DOCUMENT RESUME

ED 201 324	IR 009 296
AUTHOR	Avner, Elaine
TITLE	Summary of TUTOR Commands and System Variables. Ninth Edition.
INSTITUTION	Illinois Univ., Urbana. Computer-Based Education Research Lab.
PUB DATE	Sep 80
NOTE	186p.; For related document, see ED 154 774.
AVAILABLE FROM	University of Illinois, Computer-Based Education Research Laboratory, Urbana, IL 61801 ($7.50).
EDRS PRICE	MF01/PC08 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; Guides; Online Systems; *Programing; *Programing Languages
IDENTIFIERS	*PLATO; *TUTOR Programing Language

ABSTRACT

Intended for the experienced TUTOR author who needs a quick reference for the form of a tag and for some of the restrictions on commands, this summary does not discuss the fine details of the TUTOR language. Brief descriptions of the purpose and the tag are provided for each command. The commands are grouped in nine categories: calculating (C), data keeping (D), file operations (F), judging (J), managing sites (M), presenting (P), routing (R), sequencing (S), and terminal resident processing (T). (CHC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

ED201324

SUMMARY OF TUTOR® COMMANDS AND SYSTEM VARIABLES

(ninth edition)

Elaine Avner

PLATO Services Organization

Computer-based Education Research Laboratory

$ 7.50

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
D. Bitzer

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR009896

Copyright © September 1980
by Board of Trustees
University of Illinois

First Edition	May 1974
Second Edition	June 1975
Third Edition	November 1975
Fourth Edition	August 1976
Fifth Edition	May 1977
Sixth Edition	September 1977
Seventh Edition	July 1978
Eighth Edition	August 1979
Ninth Edition	September 1980

PLATO® and TUTOR® are service marks
of the
University of Illinois

Acknowledgment

Many PLATO users have made suggestions on the form and content of this book. I am especially grateful to members of the systems staff, the evaluation group, and the PLATO Services Organization for helpful comments. David Andersen, Brand Fortner, Dirk Grunwald, and Bruce Szablak reviewed the section on μTUTOR features.

Roy Lipschutz and Wayne Wilson assisted with final preparation of the manuscript.

This summary is intended for the experienced author who needs a quick reference for the form of a tag and for some of the restrictions on commands. It does not discuss fine details of the TUTOR language. For such information authors should refer to "aids" and to *The TUTOR Language* by Bruce Sherwood.

Each command includes a brief description of its purpose and a description of the tag. The standard form is

```
          command  brief description

                   command DESCRIPTION OF TAG   (any explanatory comments)
     or
                   command actual tag

                   Note:  Additional comments about this command.

          NOTE:  General comments about groups of commands.
```

The commands are grouped into nine categories: calculating (C), data keeping (D), file operations (F), judging (J), managing sites (M), presenting (P), routing (R), sequencing (S), terminal resident processing (T). Commands which are difficult to classify are placed in categories which describe their most probable use.

Modifications to this book required by changes in the TUTOR language are contained in lesson "aids" (press DATA and type: changes to summary). These modifications, along with other notes of interest, may be inserted in the spaces which have been provided between entries and on the additional pages at the end of each section.

CONTENTS

Page

- Abbreviations i
- Classification of commands and system variables ii to x

CALCULATING

  - Basic calculating C1
  - System operations and functions C5
  - Random numbers C8
  - Information C10
  - Bit and character manipulation C11
  - Operations on lists C14
  - Data manipulation C18
  - System variables for calculating C22

DATA KEEPING

  - Requesting data D1
  - Classifying data D2
  - Transferring data D3
  - Signing on and off D5
  - System variables for data keeping D6

FILE OPERATIONS

  - Attaching files F1
  - Datasets and namesets F2
  - Group files F7
  - TUTOR files and code files F14
  - System variables for file operations F18

JUDGING

  - Preparation for responding J1
  - Vocabulary lists J4
  - Modification of judging copy of response J5
  - Modification of judging procedure J6
  - Storing judging copy of response J8
  - Matching judging copy of response J9
  - Information on specific words in response J13
  - Unconditional judgment J15
  - Reference to other units which may contain judging commands J16
  - Alteration of judgment J17
  - Alteration of feedback J18
  - System variables for judging J19

MANAGING SITES

  - Site commands M1
  - Station commands M3

PRESENTING

  - Basic display P1
  - Graphics P6
  - Relocatable graphics P8
  - Drawing graphs P10
  - Non-screen P16
  - Special display P19
  - System variables for presenting P21

ROUTING

  - Router lesson R1
  - Curriculum information R2
  - System variables for routing R3

SEQUENCING

  - Basic sequencing S1
  - Automatic sequencing S2
  - Key-initiated sequencing S7
  - Timing S9
  - Lesson connections and sections S12
  - Lesson lists S15
  - Lesson annotation and debugging S17
  - System variables for sequencing S21

TERMINAL RESIDENT PROCESSING

  - Loading and running T1
  - Calculating T3
  - File operations T8
  - Judging T9
  - Presenting T15
  - Exchanging information with the central system T20
  - Routing T21
  - Sequencing T22
  - Running assembler programs T23
  - System variables for terminal resident processing T25

Appendix

  - Some limits associated with commands A1
  - Keyset A4
  - Keycodes, internal codes, alternate font memory locations A5
  - Powers of two A10

Alphabetical index

  - System variables I1
  - Commands I2

C
D
F
J
M
P
R
S
T
A
I

Abbreviations and Notes

Below are listed the abbreviations used in the descriptions.

abbreviation	definition
arg	argument or tag entry
(b)	blank tag
coarse	character grid coordinates
CM	central memory
CPU	central processing unit
disk	rotating magnetic disk storage unit
ECS	extended core storage
expr	mathematical expression
finex,finey	fine grid coordinates
num	number of
(opt)	optional argument
ppt	programmable terminal (IST and PPT)
string	character string
var	variable
vars	variables

In conditional statements and in statements where a variable is set, suffixes m, 0, 1, 2, etc., denote the minus condition, 0 condition, 1 condition, 2 condition, etc., e.g.,

```
match   VAR,WORD0,WORD1,WORD2
do      EXPR,NAMEM,NAME0,NAME1,NAME2
```

In conditional statements the conditional expression is rounded (not truncated) to the nearest integer. Thus, a value of -.4 results in the 0 condition being selected rather than the minus condition.

Generally, wherever a tag entry may be a number, a mathematical expression will also be accepted.

Command names are enclosed in dashes when they are referred to in descriptions, e.g., -next-. Names of system variables are enclosed in double quotes, e.g., "zreturn". Key names are capitalized, e.g., NEXT. A function key name followed by "1", e.g., NEXT1, indicates the SHIFT key is held while the key is pressed.

A word which is enclosed in single quotes designates information which is stored left-justified in a variable, e.g., 'student'.

Commands labeled "non-executable" are active only when the lesson is being condensed and not during execution.

When variables are used in the tag of certain commands which require names in the tag, e.g., -area-, the variable must be enclosed in parentheses to indicate that the information needed is the contents of the variable and not a character string; e.g., -area (v3)- means the area whose name is contained in variable v3, while -area v3- means the area whose name is v3.

CALCULATING

Basic calculating C1

define
lvars
calc
calcc
calcs
addl
subl
zero
set

Random numbers C8

seed
randu
setperm
randp
remove
restore
modperm

Information C10

clock
date
day
name
group
compute

Bit and character manipulation C11

search
pack
packc
move
itoa
otoa
htoa
clean
recname

Operations on lists C14

sort
sorta
finds
findsa
inserts
deletes
find
findall

Data manipulation C18

block
transfr
common
comload
comret
abort
commonx
initial
storage
stoload
reserve
release
backgnd
foregnd

System variables for calculating C22

lcommon
lstorag
zbpc
zbpw
zcpw
zcusers

DATA KEEPING

Requesting data D1

dataon
dataoff

Classifying data D2

area
output
outputl
setdat

Transferring data D3

readset
readd

Signing on and off D5

restart
finish
permit

System variables for datakeeping D6

collecting data

dataon

session data

zsesset
zsesspt
zsessda

area data

aarea
aarrows
ahelp
ahelpn
aok
ackist
asno
aterm
atermn
atime
auno

FILE OPERATIONS

Attaching files F1

attach
detach

Datasets and namesets F2

datain
dataout
reserve
release
setname
getname
addname
rename
addrecs
delrecs
delname
names

Group files F7

records
checkpt

TUTOR files and code files F14

setname
getname
names
iospecs
getline
setline
parse

System variables for file operations F18

zcheck
zfacc
zfile
zftype
zfusers
zinfo
zline
znindex
znscpn
znsmaxn
znsmaxr
znsnams
znsrecs
zrecs
zroff
zrstatn
zrtype
zrvars
zrvret
zsvars
zsvret
zwpb
zwpr
zxfile

JUDGING

Preparation for responding J1

eraseu
force
edit
arrow
arrowa
arheada
long
jkey
copy
endarrow

Vocabulary lists J4

list
endings
vocabs
vocab

Modification of judging copy of response J5

bump
put
putd
putv
close
loada

Modification of judging procedure J6

specs

Storing judging copy of response J8

store
storeu
storen
storea
open

Matching judging copy of response J9

match
answer
wrong
answerc
wrongc
concept
miscon
exact
exactc
exactv
ansv
wrongv
ansu
wrongu
touch
touchw
or
ans

Information on specific words in response J13

getword
getmark
getloc
compare

Unconditional judgment J15

ok
no
ignore

Reference to other units which may contain judging commands J16

join
iarrow
iarrowa

Alteration of judgment J17

judge

Alteration of feedback J18

okword
noword
markup
markupy

System variables for judging J19

judging in general

anscnt
ansok
jcount
judged
key
ntries
ztouchx
ztouchy

verbal responses

capital
entire
extra
order
phrase
spell
vocab
wcount

numerical responses

opcnt
varcnt
formok

MANAGING SITES

Site commands M1

site set
site info
site active
site stations

Station commands M3

station info
station status
station send
station logout
station stopl
station off
station on

PRESENTING

Basic display P1

at
atnm
write
writec
show
showz
showt
showe
showo
showh
showa
hidden
text
erase
lang
mode
size
rotate
delay
inhibit
char
plot

Graphics P6

dot
draw
box
fill
vector
window
circle
circleb

Relocatable graphics P8

rorigin
rat
ratnm
rdot
rdraw
rbox
rvector
rcircle

Drawing graphs P10

gorigin
axes
bounds
scalex
scaley
lscalex
lscaley
labelx
labely
markx
marky
polar
gat
gatnm
gdot
graph
gdraw
gbox
gcircle
gvector
vbar
hbar
delta
funct

Non-screen P16

slide
audio
play
record
enable
disable
ext
extout
xout
xin
beep
saylang
say
sayc

Special display P19

tabset
micro
charset
chartst
lineset
altfont

System variables
for presenting P21

mode
size
sizex
sizey
where
wherex
wherey
zlang

ROUTING

Router lesson R1

- route
- routvar
- allow

Curriculum information R2

- lesson
- score
- status

System variables for routing R3

- errtype
- ldone
- lscore
- lstatus
- rcallow
- router
- rstartl
- rstartu
- rvallow
- zleserr

SEQUENCING

Basic sequencing S1

unit
unitop
entry

Automatic sequencing S2

jump
goto
do
join
return
exit
iferror
imain
branch
doto
if
elseif
else
endif
loop
endloop
outloop
reloop

Key-initiated sequencing S7

next, nextl
back, backl
stop
nextnow
nextop, nextlop
backop, backlop
help, helpl
data, datal
lab, labl
helpop, helplop
dataop, datalop
labop, lablop
term
termop
base
end

Timing S9

keylist
pause
collect
getcode
keytype
time
timel
timer
press
catchup
break
cpulim

Lesson connections and sections S12

use
jumpout
args
from
lessin
in
notes
cstart
cstop
cstop*

Lesson lists S15

leslist
addlist
removl
reserve
release
lname
findl

Lesson annotation and debugging S17

*
c
$$
change
step
*list

System variables for sequencing S21

args
backout
baseu
clock
fromnum
key
lessnum
lleslst
llesson
mainu
mallot
muse
nhelpop
proctim
ptime
sitenam
station
tactive
user
usersin
zaccnam
zcondok
zfroml
zfromu
zgroup
zid
zlesson
zpnfile
zpnotes
zretrnu
zreturn
zsnfile
zsnotes
zsysid
zsystem
zterm
ztouchx
ztouchy
ztzone
zunit
zusers

TERMINAL RESIDENT PROCESSING

Loading and running T1

µtutor
unit
loadu
runu
haltu

Calculating T3

define
calc
calcc
calcs
zero
set
compute
randu
setperm
randp
remove
restore
block
find
pack
search
searchf

File operations T8

attach
datain
dataout

Judging T9

darrow
arrow
endarrow
long
force
jkey
copy
putd
specs
keyword
answer
wrong

Judging (cont.)

answerc
wrongc
exact
exactw
ansv
wrongv
ok
no
or
ifmatch
iarrow
ijudge
judge
getmark
getloc

Presenting T15

write
writec
show
showt
showb
showo
showh
showa
erase
mode
size
rotate
text
textn
charlim
charset
char
getchar
inhibit
allow
xout
xin
intrupt
trap

Exchanging information with central system T20

xmit
receive
sendkey

Routing T21

lesson
score

Sequencing T22

jumpn
jumpout
press
getkey
clrkey

Running assembler programs T23

pptaddr
pptload
ppttest
pptclr
pptdata
pptout
pptrun

System variables for terminal resident processing T25

zanscnt
zcomm
zdata
zentire
zextra
zjcount
zjudged
zkey
zldone
zntries
zorder
zreturn
zrouten
zscore
zspell
ztbase
ztmem
ztmemr
ztprog
ztrap
zttype
zwcount
zwherex
zwherey

Basic calculating

define (non-executable) permits an author to rename variables and to define mathematical functions, arrays, and constants for a lesson and to specify those available for student use; defined variables must physically precede any reference to the variables in the lesson

for example:

```
define  DEFNAME
        NAME1=v1,NAME2=n2,NAME3=65,NAME4=2(NAME1+NAME3),...
        FUNC(x,y,...)=some function of x, y, etc., where x, y, etc.
        are not already defined, although the expression on the right
        of the equal sign may contain previously defined names
           (up to 6 arguments are permitted)
```

The following definitions allow use of segmented variables.

```
segment,NAME=STARTING VAR,NUM BITS PER BYTE,s (opt)
segmentv,NAME=STARTING VAR,STARTING BIT POSITION,
                  NUM BITS PER BYTE,s (opt)
   (Starting variable address and byte size cannot be
   variables.  Byte size is from 1 to 59.  If the last
   argument is included, negative as well as positive integers
   may be stored.  In vertical segment, starting bit position
   may be from 1 to 60.)
```

The following definition sets up a field of less than 60 bits.

```
segmentf,NAME=VAR,STARTING BIT POSITION,NUM BITS,s (opt)
   (Restrictions are those for segmented variables.
   Field variables are not indexed.)
```

The following definitions allow use of arrays.

```
array,NAME(SIZE)=STARTING VAR   (SIZE gives number of
                                variables required)
array,NAME(NUM ROWS,NUM COLUMNS)=STARTING VAR
   (number of variables equals rows × columns)
array,NAME(FIRST ELEMENT;LAST ELEMENT)=STARTING VAR
array,NAME(FIRST ROW ELEMENT,FIRST COLUMN ELEMENT;LAST
      ROW ELEMENT,LAST COLUMN ELEMENT)=STARTING VAR
```

Arrays may also be defined with segmented variables. The form is that for vertical segments. For example:

```
arraysegv,NAME(SIZE)=STARTING VAR,STARTING BIT POSITION,
          NUM BITS PER BYTE,s (opt)
```

Up to 255 elements are permitted in an array.

(-define- continued on next page.)

```
define  student
        all defines necessary for student responses, including units

        units,UNIT1,UNIT2,...   (maximum of 10 units, such as gram,
                                meter, second,...)
           (The define set "student" may also include abbreviations
           and equivalences involving these units. See -storeu-,
           -ansu-, and -wrongu- for these applications.)
```

To merge a previous set of definitions (SETA) with a set being defined at this point in the program (SETB):

```
define  SETB,SETA
        definitions in SETB
```

To purge previous define sets:

```
define  purge,DEFNAME   (discards define set named)
define  purge   (discards all define sets except "student")
```

Note: Approximately 1500 definitions are permitted in active define sets, fewer if definitions are complicated. (Define set "student" may contain approximately 500 definitions.) Defined names and names of define sets cannot exceed 7 characters, cannot contain mathematical operators, and must start with a letter. Up to 5 define sets may be referenced. When a 6th set is activated, all earlier sets except "student" are discarded.

A local define set may be declared as a continuation of a -unit- command. (The -define- command is omitted.) Features described above are available. In addition, local variables may be declared. Sample formats for local variables are:

```
unit    someu
        NAME1,NAME2,NAME3(SIZE)
        NAME4=CONSTANT
        floating:NAME5,NAME6,NAME7(SIZE)
        integer,NUM BITS:NAME8,NAME9
        integer,NUM BITS,signed:NAME10
        integer:NAME11
```

To merge the local define set with the "global" define set:

```
unit    someu
        merge,DEFNAME   (merges with the define set named)
```

<u>or</u>

```
unit    someu
        merge,previous   (merges with the previous define set)
```

lvars (non-executable) sets up a buffer in memory for local variables for the lesson; required if the lesson uses local variables; must appear in the ieu before any references to a unit

```
lvars   SIZE OF BUFFER   (maximum size of 128)
```

calc assigns the value of the expression on the right side of the assign arrow to the variable or array on the left side, or packs up to 10 characters into an integer variable

for example:

```
calc    VAR← EXPR
calc    VAR← "STRING"   (right-justified, use n-variable)
calc    VAR← 'STRING'   (left-justified, use n-variable)
calc    ARRAYNAME← EXPR   (includes standard arithmetic operations,
           bit operations, and logical operations  on entire arrays
           and on array elements, and array functions operating on
           entire arrays)
calc    VAR← ARRAYNAME1 ∘ ARRAYNAME2
calc    ARRAYNAME1← ARRAYNAME2 X ARRAYNAME3
```

Note: See section on SEQUENCING, Automatic sequencing for -doto-, -branch-, -if-, -loop-, and related directives. These are calc-type commands which allow branching within a unit.

calcc does one of several calculations depending on the rounded value of a conditional expression

```
calcc   EXPR,VAR1← EXPRM,VAR2← EXPR0,VAR3← EXPR1,,VAR4← EXPR3
```

calcs sets a variable to one of several values depending on the rounded value of a conditional expression

```
calcs   EXPR,VAR← EXPRM,EXPR0,EXPR1,EXPR2,,EXPR4
```

NOTE: Up to 61 calculations may be performed with -calcc- or -calcs-. A blank tag entry (,,) means no calculation is done for the corresponding value of the conditional expression.

add1 adds 1 to the specified variable; can be used with array elements but not with entire arrays

```
add1    VAR
```

subl subtracts 1 from the specified variable; can be used with array elements but not with entire arrays

```
subl    VAR
```

zero sets to zero a single variable or a set of consecutive variables; can be used with array elements but not with entire arrays

```
zero    VAR
zero    STARTING VAR,NUM VARS    (cannot be used with segmented
                                  variables or segmented arrays)
```

set sets values of consecutive variables starting at the specified variable, or sets values of matrix elements starting at the specified element (starts at the first element if no element is specified); can be used to set segmented arrays but not segmented variables

```
set     STARTING VAR← VALUE1,VALUE2,VALUE3,...
set     ARRAYNAME← VALUE1,VALUE2,VALUE3,...
set     ARRAYNAME(ROW,COLUMN)← VALUE1,VALUE2,VALUE3,...
```

Note: Up to 61 values may be set with a single -set- command.

Operations and symbols used in calculations

addition	+
subtraction	-
multiplication	× or * (implied multiplication is permitted, e.g., 5a)
division	÷ or /
dot product of two arrays	∘
cross product of two arrays	X
parentheses, brackets	( ), [ ], { }
exponentiation	** or superscript or shift-superscript (e.g., a^4)
assignment of a value to a variable	←

π = pi = 3.14159...

° = degree sign; indicates a number is interpreted in degrees, e.g., 30°;
number × 1° converts number to radians;
number ÷ 1° converts number to degrees

Address of a variable may be an expression; i.e., v(EXPR) is permitted, where EXPR is rounded to the nearest integer.

Precedence of operations (in brief)

operations within parentheses

exponentiation

multiplication

division

addition and subtraction

In general with anything but very simple expressions, parentheses should be used freely.

NOTE: The computer has approximately 14-digit accuracy.
Values of floating-point numbers range from about $\pm 10^{-293}$ to $\pm 10^{+322}$.
Values of integers range from about -10^{17} to $+10^{17}$.
However, multiplication and division of large integer values may give erroneous results because of limitations on integer arithmetic.

System functions (argument may be an expression)

abs(X)	absolute value of X	
arctan(X)	inverse tangent, result in radians, range $-\pi/2$ to $+\pi/2$; for result in degrees, use arctan(X)/1°	
cos(X)	cosine of X, X in radians; use cos(X°) when X is in degrees	
exp(X)	e^X	
frac(X)	fractional part of X	X is first rounded to the nearest integer if X is within 10^{-9} of the integer for $\vert X\vert<100$ or within $(10^{-11} \times \vert X\vert)$ of the integer for $\vert X\vert>100$ (approximately)
int(X)	integer part of X	
log(X)	common logarithm of X (base 10)	
ln(X)	natural logarithm of X (base e)	
round(X)	rounded value of X	
sign(X)	$= -1$ for $X < -10^{-9}$; $= 0$ for $-10^{-9} \leq X \leq 10^{-9}$; $= +1$ for $X > 10^{-9}$	
sin(X)	sine of X, X in radians; use sin(X°) when X is in degrees	
sqrt(X)	square root of X	
varloc(X)	address of variable X (X may be a student variable, central memory variable, router variable, or defined variable)	
zfinex(X)	fine-grid x location of character-grid location "X"	
zfiney(X)	fine-grid y location of character-grid location "X"	

Logical operations and functions (logical "true" is -1; logical "false" is 0)

X = Y	equal to	equality is "true" if $\vert X-Y\vert<10^{-9}$ for $\vert X\vert<100$ or if $\vert X-Y\vert<(10^{-11} \times \vert X\vert)$ for $\vert X\vert>100$ (approximately)
X ≠ Y	not equal to	
X ≤ Y	less than or equal to	
X ≥ Y	greater than or equal to	
X < Y	less than	
X > Y	greater than	
X\$and\$Y	logical "and"; result is "true" only if both X and Y are "true"	
X\$or\$Y	logical "or"; result is "true" if either X or Y or both are "true"	
not(X)	reverse of truth value of X: if X=0, not(X)=-1; if X=-1, not(X)=0	

Bit operations and functions (use with n-variables)

X\$ars\$Y	shifts X to the right by Y bit positions	
X\$cls\$Y	shifts X to the left (circularly) by Y bit positions	
X\$mask\$Y	sets bits where bits are set in both X and Y	
X\$union\$Y	sets bits where bits are set in either X or Y or both	
X\$diff\$Y	sets bits where bits are set in either X or Y but not both	
bitcnt(X)	number of bits set in X	
comp(X)	one's complement of X (bit reversal)	
lmask(X)	left-justified number with X bits set	X ranges from 0 to "zbpw"
rmask(X)	right-justified number with X bits set	

System functions are continued on next page.

Array functions

And(X)	"true" (=-1) if all elements of array X are "true"
Max(X)	largest element in array X
Min(X)	smallest element in array X
Or(X)	"true" (=-1) if any element of array X is "true"
Prod(X)	product of all elements in array X
Rev(X)	reverse of array X; i.e., last element is now first, etc.
Sum(X)	sum of all elements in array X
Transp(X)	transpose of array X; i.e., rows and columns are interchanged

NOTE: Because of the finite accuracy of any computer, rounding occurs with operations with fractional values (v-variables), giving results which may be off by only one or two bits but which can lead to serious errors. The tolerances indicated with certain functions and logical operations are designed to avoid such problems by ignoring these least significant bits. However, there is no general solution to this inherent problem, and users must design checks for specific applications.

Some special numerical values:

1/0 = o3777 0000 0000 0000 0000
-1/0 = o4000 7777 7777 7777 7777
0/0 = o1777 0000 0000 0000 0000
-0/0 = o6000 7777 7777 7777 7777

Random numbers

seed specifies a seed for generation of random numbers with -randu- and -randp-; remains in effect until execution of another -seed- command

seed VAR CONTAINING THE SEED VALUE
seed (B) (clears former value; specifies normal system seed)

randu selects a random number, sampled with replacement, and places it in the specified variable

randu VAR,MAXIMUM (selects integer from 1 to MAXIMUM; $0 \le \text{MAXIMUM} < 2^{46}$)
randu VAR (selects a number from 0 to 1; if the tag is an n-variable, a value 0 or 1 is returned)

NOTE: The next four commands are generally used together to provide random numbers without replacement.

setperm single-argument -setperm- sets up two lists of integers from 1 to the specified value for sampling without replacement, one a working copy of the list (affected by -randp-) and the other an inactive but alterable copy of the list (affected by -remove- and -restore-); two-argument -setperm- sets up one list only; a separate copy of the list should be maintained for use with -remove- and -restore-

setperm MAXIMUM INTEGER IN LIST ($0 \le \text{MAXIMUM} \le 120$)
setperm LIST LENGTH,STARTING VAR OF LIST (for list length > 120 and for special purpose sampling; first variable of the list contains the number of elements remaining in the list and each succeeding variable contains 60 elements in the list; number of variables required is: 2+int[(length-1)/60]; length of the list is from 0 to 3000)

randp selects a random integer without replacement from the working copy of the list set up by -setperm- and places it in the specified variable

randp VAR (with 1-argument -setperm-)
randp VAR FOR STORING VALUE,STARTING VAR OF LIST (with 2-argument -setperm- or calculated list)

Note: With either form of -randp- the value returned is 0 if the working copy of the list is exhausted.

remove removes the specified value from the inactive copy of the list set up by single-argument -setperm- or from the copy of the list for two-argument -remove-

```
remove  INTEGER TO BE REMOVED   (with 1-argument -setperm-)
remove  INTEGER TO BE REMOVED,STARTING VAR OF COPY OF LIST
             (with 2-argument -setperm-)
```

restore restores the specified value to the inactive copy of the list set up by single-argument -setperm- or to the copy of the list for two-argument -restore-

```
restore INTEGER TO BE RESTORED   (with 1-argument -setperm-)
restore INTEGER TO BE RESTORED,STARTING VAR OF COPY OF LIST
             (with 2-argument -setperm-)
```

modperm (no.tag) replaces the working copy of the list with the current copy of the inactive list, which may have been operated on by -remove- and -restore-; paired with single-argument -setperm- only; to simulate -modperm- with two-argument -setperm- use -block- or -transfr- to replace the working list with the copy of the list

Information

clock	puts a character string in the specified variable giving time on a 24-hour clock in the format (space)hour.minute.second. (see also the system variable "clock") clock VAR WHERE STRING IS STORED (use -showa- to display)
date	puts a character string in the specified variable for the current date in the format (space)month number/day/last two digits of year(space) date VAR WHERE STRING IS STORED (use -showa- to display)
day	places in the specified variable the number of days elapsed since midnight December 31, 1972 to the nearest 10^{-6} day (approximately .1 second) day VAR WHERE VALUE IS STORED (use a v-variable)
name	places the signon name of the user (up to 18 characters) left-justified in two consecutive variables starting at the specified variable with octal zero fill in unused positions name STARTING VAR (requires two consecutive variables)
group	places the signon group of the user left-justified in the specified variable, up to 8 characters with octal zero fill in unused positions group VAR
compute	compiles the specified character string into machine code, executes the code, and stores the result in the specified variable; the end of the character string is determined by the specified number of characters, by six bits equal to 0 (o00), or by a comma in the string, whichever is attained first; the fourth argument is a pointer to the compiled code and is required only if the code is to be executed more than once compute VAR FOR STORING RESULT,STRING,NUM CHARACTERS IN STRING, VAR FOR POINTER TO COMPILED CODE (opt) (invalid expression does not compile; once compiled, no recompilation is done unless pointer is zeroed)

Note: STRING may be a left-justified character string enclosed in single quotes (≤10 characters) or the starting variable of a left-justified stored character string. The string may contain up to 100 characters. A "student" define set is required if the string is to be created during execution of the program.

Bit and character manipulation

search scans a character buffer for a specified object character string

search ARG1,ARG2,ARG3,ARG4,ARG5,ARG6

ARG1 = object string (left-justified)
ARG2 = number of characters in string (≤10 characters)
ARG3 = starting variable of buffer to be searched
ARG4 = total number of characters in buffer
ARG5 = relative character position at which to start search
ARG6 = variable for storing relative character position of first occurrence of object string

search ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7

ARG1 = object string (left-justified)
ARG2 = number of characters in string (≤10 characters)
ARG3 = starting variable of buffer to be searched
ARG4 = total number of characters in buffer
ARG5 = relative character position at which to start search
ARG6 = variable for storing total found count
ARG7 = number of variables following ARG6 for storing relative character positions of object string

Note: In both versions of -search- the relative found location is -1 if the object string is not found. In the second version the count is 0 if the string is not found. Relative position of the first character is 1, of the second character, 2, etc. If ARG4 (length of buffer) is negative, a backwards search is done.

pack packs a character string starting in the specified variable; the string will be left-justified with octal zero fill in unused positions; if the character count is not desired, the field is left blank

pack STARTING VAR FOR STORING STRING↕VAR FOR STORING CHARACTER COUNT↕STRING
pack STARTING VAR FOR STORING STRING↕↕STRING

Note: Use n-variable(s) for packing the string if subsequent comparison for equality with another string is done. (If the character string is packed for other purposes, v-variables are acceptable.) Segmented variables cannot be used. Embedded -show- (and related embeds) may be included in the string. Up to 500 characters (including embedded -show-) may be packed, but the tag is limited to one line of code.

packc packs one of several character strings into a variable, depending on the rounded value of a conditional expression; the string will be left-justified with octal zero fill in unused positions; if the character count is not desired, the field may be left blank; blank argument for the character string leaves the variable(s) unchanged for that value of the conditional expression

```
packc   EXPR↕STARTING VAR FOR STORING STRING↕VAR FOR STORING
        CHARACTER COUNT↕STRINGM↕STRING0↕STRING1↕↕STRING3...
```

Note: Up to 100 character strings may be listed. (See -pack- for other options and restrictions.)

move moves character(s) from a specified character position in a character string to a specified position in another character string

```
move    FROM STARTING VAR,FROM STARTING POSITION,TO STARTING
        VAR,TO STARTING POSITION,NUM CHARACTERS (opt)
move    'STRING',FROM STARTING POSITION,TO STARTING VAR,TO
        STARTING POSITION,NUM CHARACTERS (opt)
```

Note: If not specified, number of characters is 1. Maximum number of characters is 5000. Positions may be >10.

itoa converts an integer to a character string, left-justified with octal zero fill in unused positions

```
itoa    NUMBER,STARTING VAR FOR STORING STRING,VAR FOR STORING
        NUM CHARACTERS (opt)
```

Note: Non-integer values are rounded to the nearest integer before conversion to a character string.

otoa converts a number from octal format to alphanumeric format (i.e., to a character string) for the number of octal digits, if given (digits are counted from the right end of the number)

```
otoa    NUMBER,STARTING VAR FOR STORING STRING,NUM DIGITS (opt)
```

Note: Number of digits, if omitted, is 20.
If number of digits ≤10, 1 variable is used for storing the string; otherwise 2 variables are used.

htoa similar to -otoa- but for hexadecimal to alphanumeric conversion

```
htoa    NUMBER,STARTING VAR FOR STORING STRING,NUM DIGITS (opt)
```

Note: Number of digits, if omitted, is 15.
If number of digits ≤10, 1 variable is used for storing the string; otherwise 2 variables are used.

clean replaces the following character codes in a buffer with o55 (space): o00, o66 (subscript), o67 (superscript), o70 (shift), o71 (carriage return), o74 (backspace), o75 (font), o76 (access)

```
clean   STARTING VARIABLE,NUM VARS (opt)  (NUM VARS, if omitted, is 1)
```

recname takes the 18-character string specified by the first argument, removes characters not allowed in signon names, and returns the modified string in variables specified by the second argument; each string requires two variables; characters allowed are: lower-case letters, numerals, apostrophes, accent marks, asterisks, pluses, minuses, and blanks (except in the first and last position and in contiguous positions); end of the string is marked by o00

```
recname STARTING VAR OF STRING,STARTING VAR OF MODIFIED STRING
```

```
Note:  zreturn = -1  if the string is modified successfully
               =  0  if there are no characters in the initial string
               = +1  if there are no characters in the modified string
```

Note: A legal signon name must consist of at least one symbol which is a letter or numeral.

Operations on lists

sort arranges a list of entries stored in consecutive variables in ascending order according to the value of the specified sort field

```
sort    ARG1;ARG2,ARG3,ARG4,ARG5,ARG6 (opt)
        ARG1A;ARG2A   (optional line; allows simultaneous sorting
           of an associated list of entries)
```

ARG1 = starting location; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG2 = number of entries in list
ARG3 = number of variables per entry (or increment between entries); value from 1 to 200
ARG4 = starting bit position of sort field
ARG5 = number of bits in sort field
ARG6 = mask on sort field (optional)
ARG1A = starting location (see ARG1 for details) (optional)
ARG2A = number of variables per entry (optional)

Note: The field for numerical sorting may not extend across boundaries of variables.

sorta arranges a list of entries stored in consecutive variables in alphabetical order according to the internal codes for the characters in the specified sort field

```
sorta   ARG1;ARG2,ARG3,ARG4,ARG5,ARG6 (opt)
        ARG1A;ARG2A   (optional line; allows simultaneous sorting
           of an associated list of entries)
```

ARG1 = starting location; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG2 = number of entries in list
ARG3 = number of variables per entry (or increment between entries); value from 1 to 200
ARG4 = starting character position of sort field
ARG5 = number of characters in sort field
ARG6 = mask on sort field (optional)
ARG1A = starting location (see ARG1 for details) (optional)
ARG2A = number of variables per entry (optional)

Note: The field for alphabetical sorting may extend across boundaries of variables. However, the mask can affect only one variable.

finds performs a binary chop search on a sorted numerical list

finds ARG1,ARG2;ARG3,ARG4,ARG5,ARG6,ARG7,ARG8 (opt)

ARG1 = starting variable containing object of search (the object must have the same length and relative position as the list entry)
ARG2 = starting location of list; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG3 = number of entries in list
ARG4 = number of variables per entry (or increment between entries); value from 1 to 500
ARG5 = starting bit position of search field
ARG6 = number of bits in search field
ARG7 = variable for storing relative value of found location (1st entry is 1, 2nd is 2, etc); if object is not found, variable is set to negative of position where object should have been found
ARG8 = mask on search field (optional)

findsa performs a binary chop search on a sorted alphabetical list

findsa ARG1,ARG2;ARG3,ARG4,ARG5,ARG6,ARG7,ARG8 (opt)

ARG1 = starting variable containing object of search (the object must have the same length and relative position as the list entry)
ARG2 = starting location of list; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG3 = number of entries in list
ARG4 = number of variables per entry (or increment between entries); value from 1 to 500
ARG5 = starting character position of search field
ARG6 = number of characters in search field
ARG7 = variable for storing relative value of found location (1st entry is 1, 2nd is 2, etc); if object is not found, variable is set to negative of position where object should have been found
ARG8 = mask on search field (optional)

inserts inserts contents of specified buffer into a list of entries stored in consecutive variables; shifts the remainder of the list down

```
inserts ARG1,ARG2;ARG3,ARG4,ARG5,ARG6 (opt)
        ARG1A,ARG2A;ARG3A   (optional line; allows simultaneous
                             insertion into associated list)
```

ARG1 = starting variable containing object to be inserted (the object must have the same length as a list entry)
ARG2 = starting location of the list; may be:
STUDENT VAR (v or n) <u>or</u>
CM VAR (vc or nc) <u>or</u>
c,ECS COMMON LOCATION <u>or</u>
s,ECS STORAGE LOCATION
ARG3 = number of entries in the list
ARG4 = number of variables per entry (or increment between entries); ARG4 × ARG6 = value from 1 to 500
ARG5 = relative position in list at which to insert object (must be value from 1 to 1+length of list)
ARG6 = number of entries to insert (optional; default is 1)
ARG1A = starting variable containing object to be inserted (optional)
ARG2A = starting location (see ARG2 for details) (optional)
ARG3A = number of variables per entry (optional)

deletes deletes the entries at the specified position in a list of entries stored in consecutive variables; shifts the remainder of the list up and fills the last entries with zeros

```
deletes ARG1;ARG2,ARG3,ARG4,ARG5 (opt)
        ARG1A;ARG2A   (optional line; allows simultaneous deletion
                       from an associated list)
```

ARG1 = starting location; may be:
STUDENT VAR (v or n) <u>or</u>
CM VAR (vc or nc) <u>or</u>
c,ECS COMMON LOCATION <u>or</u>
s,ECS STORAGE LOCATION
ARG2 = number of entries in the list
ARG3 = number of variables per entry (or increment between entries); value from 1 to 500
ARG4 = relative position in the list of the entry to be deleted (must be a value from 1 to length of list)
ARG5 = number of entries to delete (optional; default is 1)
ARG1A = starting location (see ARG1 for details) (optional)
ARG2A = number of variables per entry (optional)

NOTE: Commands -inserts- and -deletes- may be used on sorted or unsorted lists.

find scans each variable in a set of consecutive variables for the first variable containing the specified bit pattern

```
find    ARG1,ARG2,ARG3,ARG4,ARG5 (opt),ARG6 (opt)
```

ARG1 = object bit pattern
ARG2 = starting variable in list
ARG3 = number of variables in list
ARG4 = variable for storing relative found location
ARG5 = increment between variables (optional)
ARG6 = mask (optional)

findall scans each variable in a set of consecutive variables for all variables containing the specified bit pattern

```
findall ARG1,ARG2,ARG3,ARG4,ARG5,ARG6 (opt),ARG7 (opt)
```

ARG1 = object bit pattern
ARG2 = starting variable in list
ARG3 = number of variables in list
ARG4 = variable for storing total found count
ARG5 = number of following variables for storing relative found locations (may be less than total count)
ARG6 = increment between variables (optional)
ARG7 = mask (optional)

NOTE: Use n-variables with -find- and -findall-. Segmented variables may not be used. Increment, if omitted, is 1. Negative increment causes a backward scan from the last variable in the list.
If the mask is omitted, the entire variable is compared with the object bit pattern. If a mask is used, the increment must be given, even if it is 1.
Relative position of the first variable is 0, of the second variable, 1, etc.
With -find-, if the bit pattern is not found, the found location is -1.
With -findall-, if the bit pattern is not found, the count is 0 and the first following variable is -1.

Data manipulation

block copies a set of consecutive student variables (v, n) or central memory variables (vc, nc) into another set of consecutive variables

```
block   FROM STARTING VAR,TO STARTING VAR,NUM VARS
```

transfr transfers data between v- and n-variables (student variables), vc- and nc-variables (central memory variables), ECS common, ECS storage, or vr- and nr-variables (router variables) (-comload- and/or -stoload- must be in effect with central memory variables)

```
transfr FROM STARTING LOCATION;TO STARTING LOCATION;NUM VARS
             (general form)
```

Any of the following may be used in the first two arguments of the tag:

STUDENT VAR (v or n)
common,ECS COMMON LOCATION <u>or</u> c,ECS COMMON LOCATION
storage,ECS STORAGE LOCATION <u>or</u> s,ECS STORAGE LOCATION
CM VAR (vc or nc)
router,ECS ROUTER COMMON LOCATION (when placed in "from" position, router lesson must contain -allow read-; when placed in "to" position, router lesson must contain -allow write-)
routvars,ECS ROUTER VAR LOCATION (may be placed only in "from" position; router lesson must contain -allow read rvars-)
ROUTER VAR (legal only when -transfr- command is in the router lesson)

for example:

transfr v1;c,23;10	(transfers variables v1 through v10 to ECS common locations 23 through 32)
transfr v6;vc51;9	(transfers variables v6 through v14 to variables vc51 through vc59)
transfr s,11;c,100;21	(transfers ECS storage locations 11 through 31 to ECS common locations 100 through 120)

Note: Limit to length (NUM VARS) which may be transferred is set by:
tag of -common- or -storage- (when referencing ECS) <u>or</u>
length of -comload- or -stoload- (when referencing CM variables) <u>or</u>
150 (when referencing student variables) <u>or</u>
tag of -routvar- (when referencing router variables)

NOTE: For this type of operation, -block- and -transfr- are very fast.

common (non-executable) sets up storage space which is accessible to all students in a lesson; in central memory the variables are referenced by vc and nc; common codewords must match when the common blocks are in a different lesson from the -common- command

```
common  LENGTH OF TEMPORARY COMMON,OPTIONS (opt)
common  ,COMMON NAME,LENGTH OF PERMANENT COMMON,OPTIONS (opt)
           (-common- and common blocks are in the same lesson)
common  LESSON NAME,COMMON NAME,LENGTH OF PERMANENT COMMON,
        OPTIONS (opt)
           (LESSON NAME is the lesson containing the common blocks)
```

Note: Maximum length is 8000 words. For length ≤ 1500, loading and unloading are automatic unless altered by -comload- or optional arguments "no load" or "read only". For length > 1500, -comload- must be used for access to central memory variables. Either or both of the following OPTIONS may be added to the tag of -common-:

no load (cancels automatic loading of common from ECS to CM)
read only (prevents transfer of common from CM to ECS)

The following OPTION may be used only with permanent common:

checkpt (causes common to be returned to disk approximately every 8 minutes)

comload provides automatic loading and unloading of common between central memory and ECS during each time slice; required if common length exceeds 1500 and central memory variables must be accessed

```
comload STARTING CM VAR (vc or nc),ECS COMMON LOCATION,NUM VARS
           (maximum of 1500 variables)
comload (B)   (unloads CM variables and turns off further automatic
               loading until another -comload- command is executed)
```

for example:

```
comload vc22,10,8   (transfers vc22 through vc29 from and to
                     ECS locations 10 through 17)
```

comret (no tag) returns to disk the common specified by the lesson

```
Note:  zreturn = -1  if common is successfully returned
               =  0  if no common is specified for this lesson
               = +1  if common cannot be returned
```

abort prevents return of information to disk

```
abort   common
abort   record      (with student record; sets "user" to 'sabort')
abort   autocheck   (with student record; sets "user" to 'snockpt')
abort   leslist
```

commonx similar to -common- except that -commonx- is executable; if the -commonx- command and common blocks are in different lessons, the common codewords must match or the codeword argument must be given; the codeword argument must match the common codeword on the lesson containing the blocks; if any optional arguments are included, the fields for intervening missing arguments must be present

```
commonx ,COMMON NAME,LENGTH (opt),'CODEWORD' (opt),OPTIONS (opt)
            (-commonx- and common blocks are in the same lesson)
commonx LESSON NAME,COMMON NAME,LENGTH (opt),'CODEWORD' (opt),
        OPTIONS (opt)
commonx <LESLIST POSITION>,COMMON NAME,LENGTH (opt),'CODEWORD' (opt)
        OPTIONS (opt)
commonx (B)   (disconnects current common, copies it to disk if no
              other users are in the common, and turns off comloading)
```

Note: Variable arguments must be enclosed in parentheses. Quote marks on the codeword are omitted for variable argument. If LENGTH is omitted and common is in ECS, the ECS length is used; if common is not in ECS, entire common is read from disk. OPTIONS are the same as for -common-.

```
zreturn = -1  if execution is successful
        =  0  if the common is not found or no common is in use
        =  1  if no codewords match
        =  2  if the lesson already has a common
        =  3  if the ECS version of the common has a
              different length than the declared length
        =  4  if the declared length is illegal
```

initial specifies a unit to be executed by the first user to encounter this command when a lesson or common is first brought into ECS

```
initial common,UNIT NAME   (not executed if common is in another
           lesson which is already in ECS and in which
           -initial  common- has already been executed)
initial lesson,UNIT NAME
```

storage (non-executable) sets up storage space which serves as a temporary extension of student variables; in central memory the variables are referenced by vc and nc; they are not saved in student records and are zeroed during jumpout to another lesson (see -inhibit dropstor-)

```
storage LENGTH OF STORAGE   (maximum length of 8000)
storage LENGTH,exactly   (lesson requires the exact amount of
        storage specified)
storage LENGTH,minimum   (lesson requires the greater of the amount
        of storage specified or the amount present when the lesson
        is entered with -inhibit  dropstor- in effect)
```

stoload	similar to -comload- but refers to storage; required for any length storage to access central memory variables

```
stoload STARTING CM VAR (vc or nc),ECS STORAGE LOCATION,NUM VARS
          (maximum of 1500 variables)
stoload (B)   (turns off automatic loading and unloading of storage)
```

NOTE: When both -comload- and -stoload- are used, care must be taken that the addresses of central memory variables into which common has been loaded do not overlap with the addresses into which storage has been loaded.

reserve	sets system variable "zreturn" in order to allow a user to reserve the common to prevent changes by more than one user at a time

reserve common

Note: zreturn = -2 if the common is already reserved by this user
= -1 if -reserve- is executed successfully by this user
= station number of user who has already reserved the common

release	sets system variable "zreturn" to allow the common to be released

release common

Note: zreturn = -2 if the common is not reserved by any user
= -1 if -release- is executed successfully by this user
= station number of user who has reserved the common

backgnd	(no tag) flags a lesson as a "background" lesson so that it may use large amounts of CPU time when the time is available; when CPU time is scarce, the lesson is handled at lower priority than non-background lessons
foregnd	(no tag) switches the lesson to foreground processing; normal state of execution

System variables for calculating

lcommon	length of common (set by tag of -common- command)
lstorag	length of storage (set by tag of -storage- command)
zbpc	number of bits per character (currently 6)
zbpw	number of bits per computer word (currently 60)
zcpw	number of characters per computer word (currently 10)
zcusers	number of users signed into the current common

Requesting data

dataon specifies that student data for the lesson is to be collected and sent to a datafile if the student records have been set to allow collection of data (see also system variable "dataon")

```
dataon  (B)   (turns on all data collection)
dataon  TAG   (TAG can be ok, no, unrec no, vocab, area, output,
              help, help no, term, term no, errors, signin)
```

Note: Non-blank tag will temporarily override options set in group records until turned off by -dataoff- with the appropriate tag.

dataoff specifies termination of collection of data for that lesson

```
dataoff (B)
dataoff TAG  (TAG is same as that used in a preceding -dataon-)
```

Note: Non-blank tag of -dataoff- will turn off only options turned on by a previous -dataon- with a non-blank tag; -dataoff- does not override options set in group records.

Classifying data

area specifies a section in the lesson (called an area) typically representing 5 to 15 minutes of student contact time for which certain information is collected

area NAME (maximum of 10 characters in NAME; cannot start with a number; variable tag must be enclosed in parentheses)

Note: Information collected:

elapsed time in the area (in minutes; accurate to .1 minute)
number of arrows encountered
number of "ok" judgments
number of responses judged "ok" on 1st attempt
number of anticipated "no" judgments, or "wrong" judgments
number of unanticipated "no" judgments
number of term requests satisfied
number of term requests not satisfied
number of help-type requests satisfied
number of help-type requests not satisfied
whether the area has been completed

area (B) (causes data for the preceding area to be placed in the datafile; no further data is stored until -area- with nonblank tag is executed)

area incomplete (terminates the current area and flags it as incomplete)

area cancelled (cancels all data for the current area; does not initiate a new area or produce any records in the datafile)

output puts a comment and/or value of an expression into the datafile

output COMMENT AND/OR EXPR (formats for the expression are: n, v, a, o, h with embedded form, i.e., ⟨FORMAT,EXPR⟩)

outputl places labeled information from specified student variables in the datafile

outputl LABEL (opt),STARTING VAR,NUM VARS (maximum of 10 characters in LABEL and 20 consecutive variables)

setdat allows alteration of system variables containing area data

setdat SYSTEM VAR⇐ EXPR

Note: "atime" cannot be set to a value greater than the total time signed on for that session. It is accurate to 0.1 second. The remaining system variables (except for "aarea") can be set to values up to 511.

Transferring data

readset establishes a link between a datafile and the lesson which receives the data

```
readset DATAFILE NAME,'DATAFILE CHANGE OR INSPECT CODEWORD'(opt)
        VAR FOR STORING NUM UNUSED BLOCKS (opt)
readset <LESLIST POSITION>,'DATAFILE CHANGE OR INSPECT
        CODEWORD' (opt),VAR FOR STORING NUM UNUSED BLOCKS (opt)
```

Note: The second argument is included if codewords on the lesson and datafile or do not match. Variable arguments must be enclosed in parentheses; if the codeword is a variable, quote marks are omitted. The third argument is -1 if the datafile is full.

```
zreturn = -1  if the datafile exists and is not empty
        =  0  if the name specified is not a datafile
        =  1  if codewords on the lesson and datafile do not
              match
        =  2  if the datafile is empty
        =  3  if there is no room in ECS for the buffer
        =  4  if there is a disk error
```

readd transfers data from a datafile into student variables or central memory variables (must be preceded by -readset- naming the datafile)

```
readd   area,STARTING VAR,NUM VARS
```

Note: Area summary data consists of the following information:

```
n(x) or nc(x) = starting variable
n(x) and n(x+1) contain the user's name (up to 18 characters)
n(x+2) contains the lesson name
n(x+3)      "    the area name
n(x+4)      "    elapsed time for the area (in milliseconds)
n(x+5)      "    number of arrows for the area
n(x+6)      "    number of "ok" judgments for the area
n(x+7)      "    number of "ok" judgments on the 1st attempt
n(x+8)      "    number of anticipated "no" judgments (matched by
                 -wrong-, -wrongc-, -wrongu-, -wrongv-, -miscon-,
                 -touchw-, or -judge  wrong-)
n(x+9)      "    number of unanticipated "no" judgments
n(x+10)     "    number of help-type requests satisfied
n(x+11)     "    number of help-type requests not satisfied
n(x+12)     "    number of term requests satisfied
n(x+13)     "    number of term requests not satisfied
n(x+14) = -1 if the area was completed, =0 if not
n(x+15) = -1 if the area is a continuation, =0 otherwise
```

(-readd- continued on next page)

```
readd   output1,STARTING VAR,NUM VARS
```

Note: Data from -output1- consists of the following information:

n(x) <u>or</u> nc(x) = starting variable
n(x) contains the length of -output1- (number of variables)
n(x+1) and n(x+2) contain the user's name (up to 18 characters)
n(x+3) contains the lesson name
n(x+4)   "   the area name
n(x+5)   "   execution time of -output1- in milliseconds
n(x+6)   "   -output1- label
n(x+7) to n(x + NUM VARS - 1) contain data in the tag of -output1-

```
readd   signoff,STARTING VAR,NUM VARS
```

Note: Signoff data consists of the following information:

n(x) <u>or</u> nc(x) = starting variable
n(x) and n(x+1) contain the user's name (up to 18 characters)
n(x+2) contains the lesson name
n(x+3)   "   elapsed time (in minutes) spent in the lesson during this session
n(x+4)   "   total time (in minutes) to complete the lesson if the lesson is completed during this session <u>or</u> -1 if the lesson is not completed during this session
n(x+5)   "   date of session
n(x+6)   "   time of signoff

Note: With all tags for -readd-,

zreturn = -1 if there is more data in the datafile
= 0 if the end of the datafile is reached

Signing on and off

restart specifies unit (and lesson if given) where the student is to begin at the next session

```
restart (B)   (start at the main unit containing this command)
restart UNIT NAME   (start in this lesson at the specified unit)
restart LESSON NAME,UNIT NAME   (start in the specified lesson at
                                 the specified unit)
restart <LESLIST POSITION>,UNIT NAME   (start in the lesson at the
           specified leslist position, in the specified unit;
           variable unit names must be enclosed in parentheses)
restart (0),(0)  or  restart q   (clears restart information;
                                  no restart is in effect)
```

finish specifies the unit which will be executed upon exit from the lesson via STOP1 (but not via -end lesson- or -jumpout-)

```
finish  UNIT NAME
finish  (B)  or  finish  q   (clears -finish- setting)
finish  EXPR,NAMEM,NAME0,q,NAME2,x   (example of conditional form;
           maximum of 100 arguments in the conditional tag)
```

permit specifies whether -restart- commands (except those restarting to a specific unit in another lesson) are obeyed for students in groups with short records; has no effect for students with regular records

```
permit  short recs   (permits -restart- commands except to a specific
                     unit in another lesson)
permit  (B)   (permits only -restart- commands which do not specify
              a unit; default condition)
```

System variables for data keeping

collecting data

dataon	= -1 if data collection is turned on = 0 if data collection is off (see also command -dataon-)

session data

zsesset	elapsed time since the beginning of this session (in seconds, to the nearest millisecond)
zsesspt	processing time during this session (in seconds, to the nearest millisecond)
zsessda	number of disk accesses since sign-on during this session

area data

aarea	name of current area (left-justified; display with -showa-)
aarrows	number of arrows encountered
ahelp	number of help-type requests satisfied
ahelpn	number of help-type requests not satisfied
aok	number of "ok" judgments
aokist	number of "ok" judgments on the first attempt
asno	number of specified (anticipated) "no" judgments; also referred to as "wrong" judgments, where "judged" has been set to 0
aterm	number of term requests satisfied
atermn	number of term requests not satisfied

atime	elapsed time in the area (in milliseconds)
auno	number of unanticipated "no" judgments; "judged" has been set to +1

NOTE: The system variables containing area data are zeroed at the beginning of each area.
"atime" may have a value up to about 9 hours.
The remaining variables (except "aarea") may have values up to 511.

Additional notes on DATA KEEPING

Attaching files

attach establishes connection with a file and permits access to disk records or blocks for dataset, nameset, group, TUTOR (lesson), or code files; with datasets and namesets record size may be from 64 to 512 words

```
attach  NAME   (for read and write access)
attach  <LESLIST POSITION>
attach  NAME,rw,'CODEWORD' (opt)  (for read and write access)
attach  <LESLIST POSTION>,rw,'CODEWORD' (opt)
attach  NAME,ro,'CODEWORD' (opt)  (for read-only access)
attach  <LESLIST POSITION>,ro,'CODEWORD' (opt)
```

Note: Codeword checks between lesson and file:
Codeword argument omitted, read and write access:
attach codeword on lesson must match change codeword (TUTOR file, code file) <u>or</u> write records codeword (nameset, dataset, group file);
processor lesson--read/write <u>or</u> special write or dual access (user's editing codeword [or group or account] must match the codeword for writing into the file)
Codeword argument omitted, read-only access:
attach codeword on lesson must match change or inspect codeword (TUTOR file, code file) <u>or</u> write records or read records codeword (nameset, dataset, group file);
processor lesson--read only, read/write, or dual access <u>or</u> special read or special write (user's editing codeword [or group or account] must match codeword for reading the file)
Codeword argument included: both read/write and read only:
CODEWORD (typable) must make same match as attach codeword.

Variable first or third argument must be enclosed in parentheses; quote marks on variable third argument are omitted. "rw" may be replaced by an expression with value -1. "ro" may be replaced by an expression with value 0. TUTOR files and code files are always "read only". When the file is attached with "rw", it cannot be edited by the system editor or attached with "rw" access at another station.

```
zreturn = -1  if connection to the file is successful
        =  0  if the file does not exist or is the wrong type
        =  1  if no codewords match
        =  2  if a user at another station is editing the file
              directory or has attached a TUTOR file or code
              file with "rw" access (or is editing the file)
        =  3  if there is an error in the directory of the file
        =  4  if a disk error has occurred
```

detach disconnects a file from the lesson

```
detach  NAME   (detaches and releases the specified file, whether
                active or inactive)
detach  <LESLIST POSITION>
detach  (B)    (detaches and releases the current active file)
```

Datasets and namesets

NOTE: When -datain-, -dataout-, -reserve-, -release- are used with namesets, the record designations are relative to the selected named records.

datain transfers data from disk to the desired buffer

```
datain  STARTING RECORD NUMBER;TO STARTING LOCATION;NUM
        RECORDS (opt)
```

dataout transfers data from a buffer to disk

```
dataout STARTING RECORD NUMBER;FROM STARTING LOCATION;NUM
        RECORDS (opt)
```

NOTE: With -datain- and -dataout- the second argument in the tag may be:
STUDENT VAR or c,ECS COMMON LOCATION or s,ECS STORAGE LOCATION
The number of records transferred cannot exceed the capacity of the buffer. If omitted, the number of records transferred is 1.

```
zreturn = -1  if -datain- or -dataout- is executed successfully
        =  0  if there is a pack error or if the disk containing the
              file is not available
        =  1  if the file has the wrong type, if no file is attached,
              or if no name has been selected
        =  2  if record numbers extend out of range
        =  3  if the required buffer locations extend out of range
        =  4  (-dataout- only) if the user does not have write access
        =  5  (-dataout- only) if record(s) is reserved by another user
        ≥  6  if there is a disk error
```

reserve reserves file records or the directory to prevent changes by more than one user at a time

```
reserve records,STARTING RECORD NUMBER,NUM RECORDS
reserve name   (reserves all records in the currently selected name)
reserve file   (reserves all records in the attached file)
reserve directory   (reserves the file directory)
```

```
Note:  zreturn = -2  if the records are already reserved by this user
               = -1  if -reserve- is executed successfully by this
                     user
               =  0  if no file is attached or if no name has been
                     selected
               =  1  if record numbers extend out of range
               =  2  if the user does not have write access
               =  5+n, where n=station number of the user who has
                  reserved these records
```

release releases file records or the directory

```
release records,STARTING RECORD NUMBER,NUM RECORDS
release name   (releases records in the currently selected name)
release file   (releases all records in the attached file)
release directory   (releases the file directory)
```

```
Note:  zreturn = -2  if the records are not reserved by any user
               = -1  if -release- is executed successfully by this
                     user
               =  0  if no file is attached or if no name has been
                     selected
               =  1  if record numbers extend out of range
               =  5+n, where n=station number of the user who has
                  reserved these records
```

NOTE: With the following commands (-setname-, -getname-, -addname-, -rename-, -addrecs-, -delrecs-, -delname-, -names-), the name can be up to 30 characters long (3 variables). The optional extra information for the name is stored in the right-most 24 bits of the specified variable.

setname selects a name (i.e., named set of records) in a nameset

```
setname 'NAME'  (NAME can contain up to 10 characters; if the name
            length in the nameset is more than 10 characters, the tag
            literal is filled out on the right with zeros)
setname STARTING VAR CONTAINING NAME
setname nextname   (selects the next name in alphabetical order or
            the first name if a name has not already been selected)
setname backname   (selects the preceding name in alphabetical order
            or the last name if a name has not already been selected)
setname (B)   (clears the name currently selected)
```

```
Note:  zreturn = -1  if the specified name matches exactly a name
                     in the nameset
               =  0  if the specified name matches one name for the
                     given number of characters; selects that name
               =  1  if the specified name matches more than one
                     name for the given number of characters;
                     selects the first of the names
               =  2  if the specified name does not match any name;
                     the name reference is cleared
               =  3  if no nameset is attached
```

(With tags "nextname" and "backname", "zreturn" can have only values -1, 2, or 3.)

getname stores the name currently selected and its associated extra information, if specified; the name is left-justified and unused character positions are filled with octal zeros; the extra information is stored in the right-most 24 bits of the specified variable and remaining bits are cleared

getname STARTING VAR FOR STORING NAME,VAR FOR STORING EXTRA INFORMATION (opt)

Note: If no name has been selected, a value of 0 is stored for both the name and the extra information.

addname adds a new named set of records to a nameset file; selects that name

addname STARTING VAR CONTAINING NAME,NUM RECORDS (opt),VAR CONTAINING EXTRA INFORMATION (opt)

Note: Number of records, if omitted, is 1.
Extra information, if omitted, is 0.

zreturn = -1 if the name is added successfully
= 0 if no nameset is attached
= 1 if the user does not have write access
= 2 if the new name duplicates an existing name or has an illegal format
= 3 if enough room is not available for new records
= 4 if the nameset is reserved

rename changes the name of the currently selected named set of records and/or the associated extra information

rename STARTING VAR CONTAINING NEW NAME,VAR CONTAINING NEW INFORMATION (opt)

Note: If the second argument is omitted, information is unchanged.

zreturn = -1 if the name is changed successfully
= 0 if no nameset is attached
= 1 if the user does not have write access
= 2 if no name has been selected
= 3 if the new name duplicates an existing name or has an illegal format
= 4 if records in the selected name are reserved

addrecs adds records to the selected name

```
addrecs NUM RECORDS TO ADD AT END
addrecs RECORD POSITION,NUM RECORDS TO ADD AT THAT POSITION
```

```
Note:  zreturn = -1  if records are added successfully
               =  0  if no nameset is attached
               =  1  if the user does not have write access
               =  2  if no name has been selected
               =  3  if the specified starting position is outside
                     the range of records in the named records
                     (2-argument form)
               =  4  if records in the selected name are reserved
               =  5  if enough room is not available for new records
```

delrecs deletes records from the selected name (but does not zero the records)

```
delrecs NUM RECORDS TO DELETE FROM END
delrecs STARTING RECORD POSITION,NUM RECORDS TO DELETE
```

```
Note:  zreturn = -1  if records are deleted successfully
               =  0  if no nameset is attached
               =  1  if the user does not have write access
               =  2  if no name has been selected
               =  3  if the specified starting position is outside
                     the range of records in the named records
                     (2-argument form)
               =  4  if the specified records are reserved
```

delname (no tag) deletes the currently selected name and all its records (but does not zero the deleted records)

```
Note:  zreturn = -1  if the name and records are deleted successfully
               =  0  if no nameset is attached
               =  1  if the user does not have write access
               =  2  if no name has been selected
               =  4  if the specified records are reserved
```

names stores names from the nameset list (names are stored left-justified with octal zeros in unused character positions); each name entry, which may require from 1 to 3 variables, is followed by a variable whose left-most 15 bits contain the number of records with the name and whose right-most 24 bits contain the extra information

```
names   ARG1(opt),ARG2,ARG3,ARG4

        ARG1 = starting position in list of nameset names (optional)
               (if omitted, starting position is the name currently
               selected by -setname- or at the beginning of the list
               if no name is selected)
        ARG2 = starting variable for storing names
        ARG3 = maximum number of variables for storing names (each
               requires from 2 to 4 variables)
        ARG4 = variable for storing actual number of names obtained

Note:  zreturn = -1  if names are stored successfully
               =  0  if no nameset is attached
               = +1  if the starting position is invalid
```

Group files

The following commands for namesets may also be used with group files: -setname-, -getname-, -names-, -addrecs-, -delrecs-, -reserve-, -release-, -datain-, and -dataout-.

The -records- command provides information on a group file.

NOTE: With -records- command, change access *changes* the parameter *to* the information contained in the specified variable(s); read access *stores* the parameter *in* the specified variable(s).

To change alphabetic information use a left-justified string (≤10 characters) or n-type variable(s) containing a left-justified string. To read (store) alphabetic information use n-type variables. (Alphabetic information is stored left-justified.)

records change *and* records read allow parameters for the previously selected name to be changed or read

```
records change;OPTION1;OPTION2;OPTION3...
records read;OPTION1;OPTION2;OPTION3...
```

OPTIONS:

```
name,STARTING VAR   (name of the record; requires 2 variables)
user type,VAR       (read only; stores user type, e.g., 'student')
off,VAR    (VAR is -1 for record turned off, 0 otherwise)
info,VAR   (the 24 bits of extra information)
options,'TYPE',VAR   (author and instructor options; VAR is -1 for
     option turned on, 0 for option turned off; 'TYPE' may be
     'ifilecat', 'anyless', 'sitelist', 'userlist', 'notes',
     'accounts', 'datafile', 'prints', 'editown', 'editothr')
svars,STARTING ADDRESS,STARTING VAR,NUM VARS   (student variables)
rvars,STARTING ADDRESS,STARTING VAR,NUM VARS   (router variables)
message,STARTING VAR   (requires 31 variables)
lesson,VAR       (name of the restart lesson; change:  lesson entry may
     be a leslist position, e.g., change;lesson,<LESLIST POSITION>)
unit,VAR         (name of the restart unit)
score,VAR        (last value of "lscore")
completed,VAR    (last value of "ldone")
status,VAR       (last value of "lstatus")
ldonelist,STARTING LESSON POSITION,STARTING VAR,NUM VARS  (read only;
     "ldone" information, in 3-bit signed segments; "mrouter" only)
lscorelist,STARTING LESSON POSITION,STARTING VAR,NUM VARS  (read only
     "lscore" information, in 8-bit signed segments; "mrouter" only)
data on,VAR  (VAR is -1 for individual data collection on, 0 for off)
```

(-records change- and -records read- continued on next page)

```
data opts,'TYPE',VAR   (individual data collection options; VAR is -1
     for option turned on, 0 for option turned off; 'TYPE' may be
     'area', 'output', 'ok', 'no', 'unrec no', 'vocab', 'help',
     'help no', 'term', 'term no', 'errors', 'signin')
password,TYPE,VAR (opt)   (change:  TYPE=-1, set password to string
     in VAR;  TYPE=0, zero password;  TYPE=1, set password to none;
     read:  TYPE is a variable which stores the value for type of
     password [-1, typable password; 0, blank; 1, none required])
total time,VAR   (read only; total hours on PLATO; use v-variable)
total days,VAR   (read only; total days on PLATO)
sessions,VAR     (read only; total sessions on PLATO)
cpu time,VAR     (read only; total cpu time in milliseconds)
disk count,VAR   (read only; total disk accesses)
creation,VAR     (read only; date of creation of the name)
last date,VAR    (read only; date the name was last signed on)
last time,VAR    (read only; time the name was last signed on)
station,VAR   (read only; station number where name was last signed on)
id,VAR        (read only; stores "zid" information)
```

Note: See "zreturn" on next page.

records add <u>and</u> records delete   permit names to be added or deleted

```
records add;name,STARTING VAR;user type,'USERTYPE';OPTION1;OPTION2...
records delete  (deletes the selected name and its records; zeros
                records with basic signon data but not extra records)
```

OPTIONS are those for previously selected name.
With -records add- "name" and "user type" are required; other options are allowed for initializing values. If no options are specified, the parameters are created with value of zero.
'USERTYPE' may be 'student', 'multiple', 'instructor', or 'data'. (Type 'author' may be created in author group files which do not contain University of Illinois authors.)
A 'data' user type may be used for storing data for the group as a whole. A name with this user type cannot sign onto the system.

Note: See "zreturn" on next page.

records changedir <u>and</u> records readdir   allow information in the group directory to be changed or read

```
records changedir;OPTION1;OPTION2;OPTION3...
records readdir;OPTION1;OPTION2;OPTION3...
```

OPTIONS:

```
name,VAR       (read only; name of last editor)
group,VAR      (read only; group of last editor)
```

(-records changedir- and -records readdir- continued on next page)

station,VAR (read only; station where last editor worked)
lesson,VAR (read only; lesson which last edited the group)
date,VAR (read only; date the group was last edited)
time,VAR (read only; time the group was last edited)
short,VAR (read only; VAR is -1 if the group has short records, 0 if the group has regular-length records)
snotes,VAR (read only; name of student notes file)
data file,VAR (read only; name of datafile)
processor,VAR (read only; name of processor lesson)
router,VAR (name of student router; may not be set to a system lesson; change: lesson entry may be a leslist position, e.g., changedir;router,<LESLIST POSITION>)
irouter,VAR (name of instructor router; change: if VAR=0, router is set to "imode"; may not be set to any other system lesson; change: lesson entry may be a leslist position, e.g., changedir;irouter,<LESLIST POSITION>)
group data,'TYPE',VAR (group data collection options; VAR is -1 for option turned on, 0 for option turned off; 'TYPE' is same as for individual data opts)
less data,'TYPE',VAR (lesson data collection options; VAR is -1 for option turned on, 0 for option turned off; 'TYPE' is same as for individual data opts)
write code,TYPE,VAR (opt) (write records codeword; change: TYPE=-1, set codeword to string in VAR; TYPE=1, set to unmatchable codeword; TYPE=2, set codeword to user's group; TYPE=3, set codeword to user's account;
read: TYPE is a variable which stores the value for type of codeword [-1, typable code; 1, unmatchable; 2, group; 3, account])
read code,TYPE,VAR (opt) (read records codeword; TYPE is same as for write code)
message,'TYPE',STARTING VAR,VAR FOR DATETIME (opt) (message requires 31 variables; 'TYPE'='all', 'student', 'multiple', 'instructor', or 'author'; for read-only access, date and time the message was written may be returned in the optional argument as a character string in the form: yrmndyhrmt where yr is the last 2 digits of the year; mn is month number; dy is the day of the month; hr is the hour; mt is the minute)

NOTE: For all preceding forms of the -records- command:

zreturn = -1 if -records- is executed successfully
= 0 if no group is attached or if no name has been selected
= 1 if the user does not have write access
= 2 if the new name duplicates an existing name
= 3 if there is not enough disk space available
= 4 if the entire group or the name is reserved by another user
= 5 if there is a disk error
≥ 6 if there is an error in the $(n-5)^{th}$ OPTION in the -records- tag, where n is the "zreturn" value

records locate   stores the station numbers where names in the group are signed on; the search stops when the first name is found

records locate,STATION NUMBER TO START SEARCH (opt);VAR
(if STATION NUMBER is omitted, search starts at station 0; VAR is set to the station number where the first name is found or to -1 if no names in the group are signed on at stations ≥ STATION NUMBER)

Note: zreturn = -1 if -records locate- is executed successfully
= 0 if no group is attached
= 2 if the station number is invalid
= 5 if there is a system error

records info   stores information for the name signed on at the specified station or for the previously selected name

records info,STATION NUMBER (opt);OPTION1;OPTION2;...
(if STATION NUMBER is omitted, data is stored for the selected name)

OPTIONS:

name,STARTING VAR	(name of the record; requires 2 variables)
user type,VAR	(type of record, e.g., 'student')
lesson,VAR	(name of the lesson or type of activity)
main,VAR	(name of the main unit)
current,VAR	(name of the current unit)
router,VAR	(name of the router lesson)
curriculum,VAR	(name of the curriculum file or instructor file)
zxfile,VAR	(name of the processor lesson database)
hours on,VAR	(number of hours signed on for this session; use v-variable)
cpu time,VAR	(cpu time in milliseconds for this session)
disk count,VAR	(number of disk accesses for this session)
router ecs,'TYPE',VAR	(router ecs charges; 'TYPE' is 'lesson', 'common', 'storage')
lesson ecs,'TYPE',VAR	(lesson ecs charges; 'TYPE' is 'lesson', 'common', 'storage')
area,VAR	(name of the current area)
station,VAR	(number of the station where the name is signed on)
site,VAR	(name of the site at which the name is signed on)
status,VAR	(current value of "lstatus")
score,VAR	(current value of "lscore")
completed,VAR	(current value of "ldone")

Note: zreturn = -1 if -records info- is executed successfully
= 0 if no group is attached or if no name has been selected
= 2 if the selected name is not signed on or if no name in the group is signed on at the specified station
= 5 if there is a system error
≥ 6 if there is an error in the (n-5)th OPTION in the -records- tag, where n is the "zreturn" value

records send sends a message (up to 60 characters) to the station or rings the sound device on the terminal (if programmable)

```
records send,STATION NUMBER(opt);message,SCREEN LOCATION,STARTING VAR
     CONTAINING MESSAGE,NUM CHARACTERS IN MESSAGE
records send,STATION NUMBER(opt);beep
```

(if STATION NUMBER is omitted, the message or signal is sent to the selected name)

Note: zreturn = -1 if the message or signal is sent successfully
= 0 if no group is attached or if no name has been selected
= 1 if write access to the group is not allowed
= 2 if the selected name is not signed on or if no name in the group is signed on at the specified station
= 4 if the message or signal cannot be sent to the specified station
= 5 if there is a system error

records backout backs out the specified station or the previously selected name after erasing the screen and sending a message (sets "backout" to -2)

```
records backout,STATION NUMBER (opt)
```

(if STATION NUMBER is omitted, selected name is backed off)

Note: zreturn = -1 if the backout is successful
= 0 if no group is attached or if no name has been selected
= 1 if write access to the group is not allowed
= 2 if the selected name is not signed on or if no name in the group is signed on at the specified station
= 3 if the station is not backed out because of error
= 4 if the specified station cannot be backed out
= 5 if there is a system error

records update updates the disk copy of the records for this user (student or instructor); the user's group must be attached with read/write privileges, and the user's name must be selected with -setname-

Note: zreturn = -1 if updating is successful
= 0 if no group is attached or if no name has been selected
= 1 if write access to the group is not allowed
= 2 if the wrong group is attached or the wrong name has been selected
= 3 if checkpointing has been turned off by -checkpt off- or -records save-
= 4 if the selected name is not user type 'student' or 'instructor'
= 5 if there is a disk error or other system error

records save similar to -records update- except that the record is flagged as "saved" and automatic checkpointing is turned off; can be used only in a router lesson

Note: zreturn = -1 if -records save- is successful
= 0 if no group is attached or if no name has been selected
= 1 if write access to the group is not allowed
= 2 if the wrong group is attached or the wrong name has been selected
= 3 if -checkpt off- is in effect or if the record has already been saved
= 4 if the selected name is not user type 'student' or 'instructor'
= 5 if there is a disk error or other system error

records restore retrieves the saved record from disk; used with -records save-; can be used only in a router lesson

Note: zreturn = -1 if -records restore- is successful
= 0 if no group is attached or if no name has been selected
= 1 if write access to the group is not allowed
= 2 if the wrong group is attached or the wrong name has been selected
= 3 if the record has not been saved by -records save-
= 4 if the selected name is not user type 'student' or 'instructor'
= 5 if there is a disk error or other system error

checkpt allows the program to control checkpointing of student and instructor records; a -checkpt- command executed in a router sets the default for checkpointing for subsequent instructional lessons

checkpt on	(allows automatic checkpointing; normal default for students)
checkpt off	(prevents automatic checkpointing; normal default for instructors)
checkpt EXPR	(value=0 sets to "off"; value=-1 sets to "on")

Note: zreturn = -1 if -checkpt- is successful
= 0 if -checkpt- cannot be used with this user type ('multiple', 'author', 'sabort', 'snockpt')

TUTOR files and code files

setname selects a block name from the attached file; contiguous blocks with the same name are selected at the same time

```
setname 'NAME'
setname VAR CONTAINING NAME
setname nextname   (selects the next name in sequence or the first
                   name if a name has not already been selected)
setname backname   (selects the preceding name in sequence or the
                   last name if a name has not already been selected)
setname (B)   (clears the name currently selected and selects source
              blocks set to condense)
```

Note: zreturn = -1 if the specified name matches exactly a block name in the file
= 0 if the specified name matches one name for the given number of characters; selects that name
= 1 if the specified name matches more than one name for the given number of characters; selects the first of the names
= 2 if the specified name does not match any name; the name reference is cleared
= 3 if no TUTOR file or code file is attached

(With tags "nextname" and "backname", "zreturn" can have only values -1, 2, or 3.)

getname stores the name currently selected and the associated information, if specified; name is left-justified and unused character positions are filled with octal zeros

```
getname VAR FOR STORING NAME,VAR FOR STORING INFORMATION (opt)
```

format for the information (counting from the left end of the variable):

42 bits (7 characters): block type (left-justified)
binary, charset, common, leslist, lineset, listing, micro, source, text, vocab
6 bits (1 character): condense flag [("-" (o46) or " " (o55))]
3 bits unused
9 bits: block position in directory

Note: If no name has been selected, a value of 0 is stored for both the name and the information.

names stores names of blocks in the file (left-justified with octal zeros in unused character positions); each entry requires 2 variables, the first for the name and the second for the associated information

names ARG1(opt),ARG2,ARG3,ARG4

ARG1 = starting postion in the directory of block names (numbering starts at 1 for block 1-b) (optional; if omitted, starting position is the name currently selected by -setname- or the beginning of the list if no name is selected)
ARG2 = starting variable for storing names
ARG3 = maximum number of variables for storing names (each requires 2 variables)
ARG4 = variable for storing actual number of names obtained

format for the associated information (counting from the left end of the variable):

6 bits (1 character): condense flag ["-" (o46) or " " (o55)]
6 bits (1 character): blank (o55)
12 bits (2 characters): block type

" "	(o5555)	(source)	"ll"	(o1414)	(leslist)
"bi"	(o0211)	(binary)	"ln"	(o1416)	(lineset)
"ch"	(o0310)	(charset)	"mi"	(o1511)	(micro)
"cm"	(o0315)	(common)	"tx"	(o2430)	(text)
"li"	(o1411)	(listing)	"vc"	(o2603)	(vocab)

27 bits unused (o000000000)
9 bits: number of words of disk space used

Note: zreturn = -1 if names are stored successfully
= 0 if no TUTOR file or code file is attached
= +1 if the starting position is invalid

iospecs specifies parameters for subsequent -getline- commands

iospecs OPTION1,OPTION2,OPTION3

options include:

mods	mod words are included in the lines read
nomods	mod words are not included in the lines read
deleted	deleted lines are included in the lines read
nodeleted	deleted lines are not included in the lines read
truncate	the line is truncated if it is too long for the buffer; the line pointer moves to the next line
notruncate	the line is truncated if it is too long for the buffer; the line pointer stays at the truncated line

Note: If the -iospecs- command is omitted, the default options are: nomods,nodeleted,truncate

getline reads a line from the selected block name in the attached file and stores it (left-justified) in the specified variables

getline ARG1,ARG2,ARG3

ARG1 = starting variable of the buffer for storing the line
ARG2 = number of variables in the buffer
ARG3 = variable for storing the number of variables actually required to store the line

Results depend on options set by previous -iospecs-:

mods	mod words are stored in the first 2 variables of the buffer
nomods	mod words are not stored
deleted	deleted lines are stored
nodeleted	deleted lines are not stored
truncate	a line which is too long for the buffer is truncated, and the truncated line is stored; all lines end with 12 bits of 0 (o0000); return length is the number of variables actually used to store the line
notruncate	a line which is too long for the buffer is truncated and stored (but the next -getline- command will attempt to store this line without truncation); lines so truncated will not end in 12 bits of 0; return length is the true length of the line, i.e., the number of variables that would be required to store it without truncation

Note: zreturn = -1 if -getline- is executed successfully
= 0 if no TUTOR file or code file is attached
= 1 if there are no lines left in the selected blocks
= 2 if the line length is greater than the buffer length (the line is truncated)
= 3 if this line is a deleted line
= 4 if this line is a truncated deleted line
≥ 5 if a system disk error has occurred

setline sets the pointer for the line to be read by the next -getline- command; should be used in conjunction with "zline"

setline VAR CONTAINING VALUE OF DESIRED "zline"

Note: zreturn = -1 if -setline- is executed successfully
= 0 if no TUTOR file or code file is attached
= +1 if the pointer value is illegal

parse analyzes (or parses) a line of TUTOR code stored in a buffer

`parse ARG1,ARG2,ARG3,ARG4,ARG5,ARG6`

ARG1 = starting variable containing the line of code
ARG2 = number of variables to examine (end of line also terminates the search)
ARG3 = variable for storing indent level of code (= 0 if not indented)
ARG4 = variable for storing the command name, left-justified and filled to character position 8 with spaces (o55); if the line is a comment, the first 8 characters of the line are stored
ARG5 = variable for storing the relative character position of the beginning of the tag
ARG6 = variable for storing the number of variables actually required to store this line

Note: zreturn = -1 if the line is a comment or a deleted line
= 0 otherwise

System variables for file operations

These system variables are set when the appropiate file is attached or when a name has been selected (in the attached nameset or group file).

zcheck	current checkpointing status of records = -1 if checkpointing is allowed = 0 if -checkpt off- is in effect = 1 if -records save- is in effect = 2 if -abort record- or -abort autocheck- is in effect = 3 if the user is an author or a multiple
zfacc	= -1 if the file is attached read/write = 0 if the file is attached read only or if no file is attached
zfile	name of the file currently attached to the lesson (left-justified; display with -showa-)
zftype	type of file which is attached to the lesson ('dataset', 'nameset', 'group', 'lesson', 'code') (left-justified; display with -showa-)
zfusers	number of users connected to the file currently attached
zinfo	contains the 24 bits of information associated with the currently selected name in a nameset or group; stored in the right-most 24 bits
zline	value of the pointer to the next line to be read by -getline-
znindex	position of the currently selected name in the nameset or group directory (= 0 if no name has been selected or if no nameset or group is attached)
znscpn	number of characters per name for the attached file (= 10 for TUTOR file and code file; = 18 for group file)
znsmaxn	maximum number of names (or blocks) allowed in the attached file
znsmaxr	maximum number of records (or blocks) allowed in the attached file
znsnams	number of names in use in the attached nameset or group

zn**s**recs	number of records in use in the entire attached nameset or group
zrecs	number of records in the selected name in the attached nameset or number of extra records (i.e., records added with -addrecs-) in the selected name in the attached group or number of records in the attached dataset
zroff	= -1 if the currently selected name in a group has been turned off = 0 otherwise
zrstatn	station number where the currently selected name in a group is signed on (= -1 if the name is not signed on)
zrtype	user type of currently selected name in the attached group: 'student', 'multiple', 'instructor', 'author', 'data' (left-justified; display with -showa-)
zrvars	maximum number of router variables (currently 64)
zrvret	= -1 if router variables are permanently stored on disk = 0 if router variables are not permanently stored
zsvars	maximum number of student variables (currently 150)
zsvret	= -1 if student variables are permanently stored on disk = 0 if student variables are not permanently stored
zwpb	number of computer words per block in the attached TUTOR file or code file (currently 320)
zwpr	number of computer words per record in the attached file (= 320 for TUTOR file and code file; = 64 for group file)
zxfile	contains the name of the file through which a processor lesson is accessed (= 0 if the processor lesson is entered directly) (left-justified; display with -showa-)

Additional notes on FILE OPERATIONS

Two types of commands are described in this section: judging commands and regular commands. (In all other sections of this book only regular commands are described.) Regular commands are not executed during the judging process, i.e., after the student has entered a response, nor are judging commands executed before the student has entered a response or in situations where no response is involved. (See "The TUTOR Language" and lesson "aids" for extensive descriptions of the judging process.)

Regular commands in this section include: -eraseu-, -force-, -edit-, -arrow-, -arrowa-, -arheada-, -long-, -jkey-, -copy-, -endarrow-, -getword-, -getmark-, -getloc-, -compare-, -iarrow-, -iarrowa-, -judge-, -okword-, -noword-, -markup-, -markupy-. Commands -list-, -endings-, -vocabs-, -vocab- are special non-executable commands which establish lists of words for use with certain response-matching commands. The -join- command is both regular and judging. The remaining commands in this section are judging commands.

Preparation for responding

eraseu (regular command) the specified unit is executed at all subsequent arrows in the unit containing -eraseu- when the student erases part or all of a response after receiving judgment; useful for erasing complicated displays which are not handled by the standard judging process

```
eraseu  UNIT NAME
eraseu  (B) or  eraseu  q    (clears -eraseu- setting for remainder
                             of the unit)
eraseu  EXPR,NAMEM,NAME0,q,NAME2,x   (example of conditional form;
             maximum of 100 arguments in the conditional tag)
```

force (regular command) alters the input of a response as specified; setting is cleared at each main unit

```
force   firsterase   (erases an incorrect response and contingent
             message when the student presses any key, not just
             NEXT, ERASE, etc.)
force   font   (inserts the font code before the first keypress)
force   left   (forces the response to be written from right to left)
force   long   (initiates judging when the character input reaches
             the value of the tag of -long-; unnecessary with -long  1-)
force   micro   (forces all keypresses through the microtable
                 conversion)
force   (B) or  force   clear   (clears previous -force- settings
                                 in that unit)
force   clear,font,left   (may combine tags)
```

edit (regular command) required for EDIT key to be active when the tag of -long- exceeds 150; specifies the starting variable of a buffer for storing the characters in a response (up to 300 characters)

```
edit    STARTING VAR   (use student variable)
edit    (B)   (clears edit buffer; if placed after -arrow-, prevents
              use of the EDIT key; see also -inhibit  edit-)
```

arrow (regular command) plots the response arrow at the specified screen location (see -inhibit arrow-); sets defaults: -long 150- and -jkey (B)-

```
arrow   COARSE
arrow   FINEX,FINEY
```

arrowa (regular command) allows an alternative arrow associated with -iarrowa- and -arheada-; follows same rules and restrictions as -arrow-

```
arrowa  COARSE
arrowa  FINEX,FINEY
```

arheada (regular command) specifies a symbol to be plotted with the alternative arrow

```
arheada SYMBOL TO BE PLOTTED WITH -arrowa-
```

Note: Up to five 6-bit characters may be specified.

long (regular command) sets the maximum number of characters in a response (default is 150 characters); must follow -arrow- (see NOTE)

```
long    NUM CHARACTERS   (value of tag is from 0 to 300; -long  1-
           causes automatic judging; tag > 150 requires use of
           -edit- for EDIT key to be active; -long  0- prevents
           input from the keyset except for function keys)
```

jkey (regular command) specifies the function key(s) which will initiate judging (in addition to the NEXT key); must follow -arrow- (see NOTE)

```
jkey    KEYNAME   (name of function key is in lower case)
jkey    KEYNAME1,KEYNAME2,KEYNAME3,...
jkey    (B)   (clears previous -jkey- settings so that only NEXT
              initiates judging)
```

copy (regular command) specifies the starting variable of the character string which is to be copied on the screen at the arrow, one word at a time, when the COPY key is pressed; the end of the character string is indicated by the specified number of characters or by 12 bits equal to 0 (o0000), whichever is attained first; loads the sting as it appears on the screen into the response buffer; must follow -arrow- (see NOTE)

```
copy    STARTING VAR,NUM CHARACTERS   (use student variable)
```

NOTE: To affect the first response, -long-, -jkey-, and -copy- must follow the -arrow- command but must precede any judging commands. However, after the student enters a response (e.g., an incorrect response), these commands can be executed among the regular commands following the matched response in order to affect the next response at the same arrow.

endarrow (regular command) (no tag) terminates judging with an unanticipated "no" judgment if the response has not been matched; after an "ok" judgment, -endarrow- terminates the search for additional -arrow- commands and switches back to the pre-arrow state

Vocabulary lists

list (non-executable) sets up a list of synonyms for judging; used with -answer-, -wrong-, -answerc-, -wrongc-, and -match-

```
list    LISTNAME,WORD1,PHRASE*CONSISTING*OF*SEVERAL*WORDS,
        WORD2,WORD3,...   (maximum of 7 characters in LISTNAME)
```

endings (non-executable) used with -vocabs- and -vocab- to add endings to root words (must precede -vocabs- or -vocab-)

```
endings NUMBER,ENDING1,ENDING2,...  (NUMBER is an integer from 0 to 9)
```

Note: In -vocabs- or -vocab-,

WORD/NUMBER adds endings to the root word and includes all words in the vocabulary

WORD//NUMBER adds only words with endings to the vocabulary; the root word is not included

Up to 10 -endings- commands with up to 8 endings each may be included. Apostrophe is legal in an ending.

vocabs (non-executable) sets up lists of ignorable words and synonymous required words; used with -concept- and -miscon-; checks for capitalization and spelling; allows assignment of user information numbers

```
vocabs  NAME
        <IGNORABLE WORDS SEPARATED BY COMMAS>
        WORD1,WORD2,PHRASE*CONSISTING*OF*SEVERAL*WORDS
        (SYNONYMOUS WORDS3 AND PHRASES SEPARATED BY COMMAS)
        (WORD4/s,WORD5/ENDING1/ENDING2,WORD6//ENDING1)
        WORD7/NUMBER1,WORD8//NUMBER2
        (WORD9,WORD10=1,WORD11=2,SYNONYM11=2,...)
```

vocab (non-executable) similar to -vocabs- except does not check for capitalization and spelling and does not allow phrases

```
vocab   NAME
        <IGNORABLE WORDS SEPARATED BY COMMAS>
        WORD1,WORD2
        (SYNONYMOUS WORDS3 SEPARATED BY COMMAS)
        (WORD4/s,WORD5/ENDING1/ENDING2,WORD6//ENDING1)
        WORD7/NUMBER1,WORD8//NUMBER2
```

NOTE: Up to 7 characters are permitted in the name of the vocabulary. When sets of endings are used repeatedly, -endings- plus -vocab(s)- may be more convenient than -vocab(s)- with actual endings included. With -vocabs-, user information numbers may have values from 1 to 511.

Modification of judging copy of response

bump removes the specified characters from the judging copy of the response before judging

```
bump    CHARACTERS    (maximum of 8 characters; use additional
                      -bump- commands for more than 8 characters)
```

put replaces a character string in the judging copy of the response with another character string

```
put     STRING1=STRING2    (replaces STRING1 with STRING2)
```

putd similar to -put- but uses the first character in the tag as the separator between strings

```
putd    /STRING1/STRING2/    (separator is /)
putd    ,STRING1,STRING2,    (separator is ,)
```

putv similar to -put- but works with stored strings

```
putv    ARG1,ARG2,ARG3,ARG4

        ARG1 = starting variable of string (left-justified)
        ARG2 = number of characters in string
        ARG3 = starting variable of replacement string (left-justified)
        ARG4 = number of characters in replacement string
```

NOTE: Maximum number of characters in a string for -put-, -putd-, and -putv- is 50. If replacement operations cause the judging copy of the response to exceed 300 characters, judging terminates with a "no" judgment.

close takes characters stored in the right-most six bits from successive variables and makes a judging copy for use with judging commands; often paired with -open-; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (o00), whichever is attained first

```
close   STARTING VAR,NUM CHARACTERS    (use n-variables)
```

loada takes the characters stored in the specified variable(s) by -pack-, -storea-, or -calc- and makes a judging copy; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (o00), whichever is attained first

```
loada   STARTING VAR,NUM CHARACTERS (opt)    (number of characters,
              if omitted, is 10; maximum number of characters is 299)
```

Modification of judging procedure

NOTE: The various -specs- options do not affect all judging commands. Commands affected by each -specs- option are indicated by number from the following list.

judging commands affected by -specs-

1. -match-
2. -answer-, -wrong-, -answerc-, -wrongc-
3. -vocabs-, -concept-, -miscon-
4. -vocab-, -concept-, -miscon-
5. -exact-, -exactc-, -exactv-
6. -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-
7. -storen-
8. -storea-

specs allows control over processing of responses; also serves as a marker for execution of subsequent regular commands after judging is complete

specs allwords (treats integers like letters [rather than numbers] so that a number-letter boundary is not like a word-word boundary or punctuation)
(with 1, 2, 3, 4, 7 above)

specs alphxnum (treats a letter-number boundary like a word-word boundary or like punctuation)
(with 1, 2, 3, 4, 7 above)

specs bumpshift (removes shift codes from the judging copy of the response)
(with all commands above)

specs exorder (checks the order of ignorable words)
(with 2 above)

specs holdmark (prevents markup of student response but stores the markup information for later display)
(with all commands above where markup is done: 2, 3, 4)

specs nodiff (turns off the numeric difference judger, which treats a numerical response as a "misspelling" if it is within 10% of the correct response; no spelling markup is done)
(with 2 above)

specs nomark (turns off answer markup)
(with all commands above where markup is done: 2, 3, 4)

specs nookno (prevents appearance of "ok" and "no")
(with all commands above)

```
specs   noops       (prevents use of mathematical operations in a
                    numerical response)
        (with 6 above)

specs   noorder     (turns off the order judger; allows any
                    word order; no order or keyword markup is done)
        (with 2, 3, 4 above)

specs   nospell     (turns off the spelling judger; no spelling
                    markup is done)
        (with 2 above)

specs   novars      (prevents use of variables defined in
                    define set "student")
        (with 6 above)

specs   okassign    (allows assignment of a value to a variable
                    defined in define set "student")
        (with 6 above)

specs   okcap       (allows a capitalized word in the response to
                    match a non-capitalized word in the tag of a
                    response-matching command or in the vocabulary)
        (with 1, 2, 3 above)

specs   okextra     (allows extra words in the response;
                    caution--words not in -vocabs- may be
                    treated as spelling errors)
        (with 2, 3, 4 above)

specs   okspell     (allows any reasonable spelling)
        (with 1, 2, 3 above)

specs   toler       (allows 1% tolerance in a numerical response)
        (with 1, 2 above)

specs   (B)         (acts only as a marker)

specs   nookno,okspell,noorder   (may combine tags)
```

Note: The following system variables are set properly even if use of the -specs- tag causes the response to match the tag of a response-matching command.

-specs- tag	system variable
okspell	spell
okcap	capital
okextra	extra
noorder	order

Storing judging copy of response

NOTE: Commands -store-, -storeu-, and -storen- terminate judging with a "no" judgment if an error is found in the form of the response.

store calculates the numerical value of the response and stores it in the variable specified in the tag

```
store   VAR
```

storeu similar to -store- but also evaluates dimensionality of units

```
storeu  VAR FOR STORING NUMBER,STARTING VAR FOR STORING POWERS
        OF DIMENSIONS   (see -define-: units; use v-variables to
           store the powers of the dimensions; powers are stored
           in the order in which the primary units are defined)
```

storen searches for and evaluates simple numerical expressions (without variables) in the response, which may also contain non-numeric characters; stores numerical results in the specified variables one at a time; removes numerical parts of the response from the answer buffer and replaces them with blanks; numerical parts must be set off from letters by spaces or punctuation; if no numerical expression is found, the variables are set to 0 and judging ends with a "no" judgment; each -storen- increments "anscnt"

```
storen  VAR1
storen  VAR2
   .
   .
   .
```

storea stores characters from the response, left-justified, in the specified variable(s), 10 characters per variable; unused character positions are filled with octal zeros

```
storea  STARTING VAR,NUM CHARACTERS (opt)   (number of characters,
                                             if omitted, is 10)
```

Note: Use n-variable(s) for storing the string if subsequent comparison for equality with another string is done. (If the character string is stored for other purposes, v-variables are acceptable.) Segmented variables cannot be used with -storea-.

open places the characters in the response, one-by-one, in the right-most six bits of successive variables starting at the specified variable

```
open    STARTING VAR   (use n-variables)
```

Matching judging copy of response

NOTE: In this sub-section references to response mean judging copy of the response.

NOTE: Up to 39 "words" (entries separated by space or punctuation) are permitted in responses with -match-, -answer-, -wrong-, -answerc-, -wrongc-, -concept-, and -miscon-. If the number of words exceeds 39, judgment is "no" and "anscnt" is set to -2.

match checks the response against the arguments in the tag and sets a variable to the relative position of the matched character string; if no match is found, the variable is set to -1 and judgment is "no" ("judged" set to +1); otherwise judgment is "ok" ("judged" set to -1); always ends judging

for example:

```
match   VAR,WORD0,WORD1,PHRASE*WORD2,WORD3
match   VAR,(WORD0,SYNONYM0),(WORD1,(LISTNAME1)),((LISTNAME2))
```

answer compares the response with the tag; checks for word order, spelling, capitalization, extra words, and numeric tolerance unless altered by -specs-; sets "judged" to -1 if response matches tag

for example:

```
answer  WORDS AND PHRASE*WORDS   (blank tag matches a response in
            which nothing is entered or which contains only spaces
            and punctuation)
answer  <EXTRA WORDS>(SYNONYMOUS WORDS1 AND PHRASE*WORDS1
        SEPARATED BY COMMAS)(SYNONYMOUS WORDS2 and PHRASE*WORDS2
        SEPARATED BY COMMAS)WORD3
answer  <<LISTNAME1>>((LISTNAME2),WORD2)
answer  RESPONSE1;RESPONSE2;RESPONSE3   (each argument may have any
            of the preceding forms for the tag of -answer-; synonymous
            responses for the same argument are separated by commas)
```

Note: Tag must not contain punctuation (or symbols changed to "punc" in the -change- command), although the student's response may.

wrong compares the response with the tag; performs checks available with -answer-; sets "judged" to 0 if response matches tag

```
wrong   WORDS AND PHRASE*WORDS
```

Note: Options for the tag of -wrong- are the same as for the tag of -answer- but for an incorrect student response.

answerc compares the response with one of several arguments in the tag, depending on the rounded value of the conditional expression; performs the checks available with -answer-; sets "judged" to -1 if response matches the required argument

```
answerc EXPR↕RESPONSEM↕RESPONSE0↕↕RESPONSE2   (the arguments may
              have any of the forms allowed in the tag of -answer-;
              a blank argument indicates no anticipated response for
              that value of the conditional expression)
```

wrongc same options as for -answerc- but for an incorrect response; sets "judged" to 0 if response matches the required argument

```
wrongc  EXPR↕RESPONSEM↕↕↕RESPONSE2↕RESPONSE3
```

concept compares the response with the tag; -vocab- or -vocabs- provides synonyms; sets "judged" to -1 if response matches tag

```
concept WORDS AND PHRASE WORDS   (no asterisk in phrases; blank tag
              matches a response which is blank or which contains only
              ignorable words from the vocabulary)
concept WORD1 WORD2,VAR1⇐WORD1,VAR2⇐WORD2   (detects which synonym
              is entered if the vocabulary is appropriately set up)
```

miscon same options as for -concept- but for an incorrect response; sets "judged" to 0 if response matches tag

```
miscon  WORDS AND PHRASE WORDS
```

exact compares the response with the tag for an exact character string match; sets "judged" to -1 if response matches tag

```
exact   STRING   (blank tag matches a response in which nothing
                 is entered)
```

exactc compares the response for an exact character string match with one of several arguments in the tag depending on the rounded value of a conditional expression; sets "judged" to -1 if response matches the required argument

```
exactc  EXPR,STRINGM,STRING0,,STRING2   (blank argument matches a
              response in which nothing is entered)
```

exactv compares the response with a stored character string for an exact match; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (o00), whichever is attained first; sets "judged" to -1 if response matches tag

exactv STARTING VAR OF STORED STRING,NUM CHARACTERS (opt)
(number of characters, if omitted, is 10; if the number of characters is 0, the response is judged correct if nothing is entered)

NOTE: With the following four commands (-ansv-, -wrongv-, -ansu-, -wrongu-), TOLERANCE is optional. When tolerance is omitted, the default is 10^{-9} if the absolute value of the tag value is less than approximately 100 or ($10^{-11} \times |\text{tag value}|$) if the absolute value of the tag value is greater than approximately 100. These commands cannot judge values smaller in absolute value than 10^{-9} since any response less than 10^{-9} will then match the tag.
TOLERANCE may be absolute deviation or percent deviation.

ansv checks a numerical response against the first argument in the tag, with tolerance, if given, set by the second argument; sets "judged" to -1 if response matches tag (within the tolerance)

ansv CORRECT VALUE,TOLERANCE

wrongv similar to -ansv- but for the incorrect numerical response; sets "judged" to 0 if response matches tag (within the tolerance)

wrongv INCORRECT VALUE,TOLERANCE

ansu similar to -ansv- but checks for correct units; sets "judged" to -1 if response matches tag (within the tolerance)

ansu NUMBER AND UNITS,TOLERANCE

wrongu similar to -ansu- but for incorrect response; sets "judged" to 0 if response matches tag (within the tolerance)

wrongu NUMBER AND UNITS,TOLERANCE
wrongu NUMBER,TOLERANCE (may be used to indicate that units are missing)

NOTE: For applications of -ansu- and -wrongu- see -storeu- and -define-. Commands -ansv- and -wrongv- accept defined units in the student's response but do not compare with author's units.

touch specifies the location of a rectangle for a touch response; sets "judged" to -1 if the specified rectangle is touched (see -enable- and -disable-)

```
touch   AREA1;AREA2;AREA3;...  (blank tag matches any touch input)
```

Note: AREA may be: COARSE,WIDTH IN CHARACTERS,HEIGHT IN LINES
<u>or</u> FINEX,FINEY,WIDTH IN DOTS,HEIGHT IN DOTS
COARSE or FINEX,FINEY is the screen location of the lower left corner of a rectangle of specified width and height. Width and height are optional and assumed to be 1 if omitted.

touchw same options as -touch- but for an incorrect touch response; sets "judged" to 0 if the specified rectangle is touched

```
touchw  AREA1;AREA2;AREA3;AREA4;...
```

NOTE: One touch square is 32 dots on each side (or 4 characters in width and 2 lines in height).

or (no tag) placed on the line between response-matching commands to provide alternative responses for the same value of "anscnt"

ans (no tag) allows use of the ANS key; terminates judging only if ANS is pressed; otherwise normal judging occurs; -ans- must be the first judging command following the -arrow- command unless -jkey ans- is in effect

Information on specific words in response

NOTE: In the following commands (-getword-, -getmark-, -getloc-, -compare-), a "word" is an entry set off by spaces or punctuation from surrounding characters. (See -specs allwords-, -specs -alphxnum- and -change symbol- for additional options in specifying word boundaries.)

getword (regular command) allows storage of individual words in a response

```
getword ARG1,ARG2,ARG3,ARG4 (opt)
```

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = starting variable for storing the word (up to 10 characters per variable)
ARG3 = variable for storing the actual number of characters in the word (=0 if ARG1>"wcount")
ARG4 = maximum number of characters to be stored in ARG2 (optional; if omitted, value is 10)

Note: Words that are stored are not removed from the judge buffer.

getmark (regular command) used after judging a response to give markup information on individual words in the response

```
getmark ARG1,ARG2
```

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable containing markup information
- = -2 if the response is perfect or if no markup is done with the response-matching command used
- = -1 if the position of the word is out of bounds (i.e., if ARG1>"wcount")
- = 0 if there are no errors in the word
- > 0 bits in ARG2 are set according to the error(s), starting at the right-most bit (subscript "2" indicates the number is in binary notation):
  - (1_2) a word preceding this word is missing
  - (10_2) the word is out of order (too far right)
  - (100_2) there is a capitalization error
  - $(1\ 000_2)$ the spelling is incorrect
  - $(10\ 000_2)$ the word is part of a broken phrase
  - $(100\ 000_2)$ the word is an extra word
  - $(1\ 000\ 000_2)$ this word is the last word, and a word which should follow is missing

getloc (regular command) gives the screen position of the beginning (and end, if requested) of the specified word in the response

```
getloc  ARG1,ARG2,ARG3,ARG4 (opt),ARG5 (opt)
```

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable for storing the finex screen position of the beginning of the word (= -1 if ARG1 > "wcount")
ARG3 = variable for storing the finey screen position of the beginning of the word
ARG4 = variable for storing the finex screen position of the end of the word (optional)
ARG5 = variable for storing the finey screen position of the end of the word (optional)

compare (regular command) compares two words for spelling

```
compare AUTHOR WORD,STUDENT WORD,VAR FOR STORING RESULT CODE
```

```
result = -1  if words are different
       =  0  if words are identical
       >  0  if words are misspellings of each other
             (smaller value indicates a closer match)
```

Note: The first two arguments may also be variables. The words must be stored in the variables in the same manner, e.g., both words left-justified or both right-justified. If the words are stored with -storea- or -pack-, they will be left-justified. If a word itself is given, rather than the variable(s), it should be enclosed in single quotes for left-justification or double quotes for right-justification.
System variables "spell" and "capital" are set appropriately if result value ≥ 0:
"spell" is set to 0 if result > 0;
"capital" is set to 0 if only one word is capitalized

Unconditional judgment

ok	(no tag) judges any response "ok"; sets "judged" to -1
no	(no tag) judges any response "no"; sets "judged" to +1
ignore	(no tag) erases and ignores any response; stops further processing and waits for a new response

Reference to other units which may contain judging commands

join causes execution of the specified unit without change of main unit; commands following -join- are executed; -join- is executed in all states: regular, judging, and search (see also description under SEQUENCING, Automatic sequencing)

```
join    UNIT NAME
join    NAME,VAR⇐INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
join    EXPR,NAMEM,NAME0,x,NAME2,q   (example of conditional form)
```

iarrow (regular command) the specified unit is executed after each subsequent -arrow- in a unit just before the first judging command is executed

```
iarrow  UNIT NAME
iarrow  (B)  or  iarrow  q   (clears the -iarrow- setting for
            subsequent -arrow- commands in the unit)
iarrow  EXPR,NAMEM,NAME0,q,NAME2,x   (example of conditional form;
            maximum of 100 arguments in the conditional tag)
```

Note: The -iarrow- setting is equivalent to -join- after each subsequent -arrow- (just before the first judging command); the specified unit is executed in all states.

iarrowa (regular command) similar to -iarrow- but is associated with -arrowa-; see -iarrow- for restrictions

```
iarrowa UNIT NAME
```

Alteration of judgment

judge (regular command) alters the judgment rendered by judging commands

judge	ok	(sets judgment to "ok"; sets "judged" to -1; continues processing regular commands)
judge	no	(sets judgment to "no" [unanticipated]; sets "judged" to +1; continues processing regular commands)
judge	wrong	(sets judgment to "no" [anticipated]; sets "judged" to 0; continues processing regular commands)
judge	okquit	(sets judgment to "ok"; sets "judged" to -1; terminates processing at that arrow except for regular commands following -specs-)
judge	noquit	(sets judgment to "no"; sets "judged" to +1; terminates processing at that arrow except for regular commands following -specs-)
judge	quit	(leaves judgment unchanged and terminates processing at that arrow except for regular commands following -specs-; allows the student to proceed to the next arrow even if judgment on the current arrow is "no")
judge	ignore	(stops all processing, erases the response, and waits for a new response)
judge	exit	(rescinds judgment and waits for additional keys)
judge	continue	(resumes judging using the modified response, as altered by -bump-, -put-, -specs-, -match-, -storen-, etc.; resumes processing judging commands)
judge	rejudge	(resumes judging using the original, unmodified response and clears the -specs- setting; resumes processing judging commands)
judge	EXPR,x,no,ignore,ok	(example of conditional form; argument x leaves judgment unchanged)

Alteration of feedback

okword (regular command) permits "ok" message to be changed

```
okword  NEW WORD FOR USE WITH "OK" JUDGMENT   (may be blank)
```

noword (regular command) permits "no" message to be changed

```
noword  NEW WORD FOR USE WITH "NO" JUDGMENT   (may be blank)
```

NOTE: Tags of -okword- and -noword- may have up to 9 characters. A space is automatically provided before the message. Commands -okword- and/or -noword- may be placed anywhere in the lesson. Once they are executed, they are in effect until execution of another -okword- and/or -noword- command.

markup (regular command) (no tag) used with -specs holdmark- to display markup information that was inhibited with -specs holdmark-

markupy (regular command) specifies vertical displacement of markup information in screen dots from the default position of 8 dots below the response; tag is negative for new position below the default, positive for above; new position is in effect until execution of another -markupy- command

```
markupy DOTS FROM DEFAULT MARKUP POSITION
markupy 0   (set to default position of 8 dots below response)
```

System variables for judging

judging in general

ansent	number of response-matching commands encountered at an arrow before the response is matched; also set by -storen-; otherwise, = -2 if the student's response contains more than 39 words = -1 if no tag is matched = 0 for a store error zeroed for each -arrow- and each -specs- command
ansok	= -1 if the response is a satisfactory match to the preceding response-matching command = 0 otherwise; in particular, after -no-, = -1 if there is no match to a previous response-matching command = 0 if the match is poor
jcount	number of internal 6-bit character codes in the response
judged	= -1 for any "ok" judgment = 0 for any "wrong" judgment (anticipated "no") = 1 for any "no" judgment (unanticipated "no") = 2 for response not yet judged
key ztouchx ztouchy	[See descriptions under system variables for sequencing.]
ntries	number of attempts on the current arrow

verbal responses

judging commands which affect system variables for verbal responses:

1. -match-
2. -answer-, -wrong-, -answerc-, -wrongc-
3. -vocabs-, -concept-, -miscon-
4. -vocab-, -concept-, -miscon-

capital	= -1 if there are no capitalization errors, = 0 otherwise (with 1, 2, 3 above)
entire	= -1 if all required words are present, = 0 otherwise (with 2 above)

extra	= -1 if there are no extra words in the response, = 0 otherwise (with 2, 3, 4 above)
order	= -1 if word order is ok, = 0 otherwise (with 2 above)
phrase	= -1 if there are no phrase errors, = 0 otherwise (with 1, 2, 3 above)
spell	= -1 if spelling is ok, = 0 otherwise (with 1, 2, 3 above)
vocab	= -1 if all words in the response are in the vocabulary, = 0 otherwise (with 3, 4 above)
wcount	number of words in the response (maximum of 39) (with all above)

numerical responses

These system variables are set with -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-, -compute-, and the -calc- type commands.

opcnt	number of arithmetic operations and functions in the response (= -1 if there are no operations and the expression cannot be stored with -store-)
varcnt	number of defined variables and functions (define set "student")

formok gives diagnostics on errors in mathematical expressions

= -1 if the expression is ok
= 0 if there is a bad function argument or index
= 1 if there is an illegal character
= 2 if there are unbalanced parentheses
= 3 if there are too many decimal points
= 4 if there are variables not defined in define set "student"
= 5 if a symbol involving $ is not a logical or a bit operator
= 6 if the expression has bad form
= 7 if a value is assigned to a non-variable
= 8 if an octal constant contains digit 8 or 9
= 9 if there is an error in an alpha string
= 10 if a number has too many digits
= 11 if an array index is out of bounds
= 12 if there are variables with -specs novars-
= 13 if there are operations with -specs noops-
= 14 if there are assignments without -specs okassign-
= 15 if units in the response are used incorrectly
= 16 if too much computing is attempted
= 17 if the expression is too deep in nested functions
= 18 if a function has the wrong number of arguments
= 20 if an array has the incorrect number of arguments
= 21 if an array is not permitted in this expression
= 22 if the array size is incorrect or operation is nonconformal
= 23 if there are too many arrays in the expression
= 60 if too many temporary variables are needed during processing
= 62 if expression is too complicated for temporary storage to hold
= 63 if there are too many literal numbers in the expression
= 65 if there is an error in a segment definition
= 66 if expression is too deep in indexes which are assigned values

Additional notes on JUDGING

Commands in this section are legal only in a site lesson.

Site commands

These commands, all of which have -site- in the command field, give information on the specified logical site.

site	set	specifies the logical site for subsequent -site- commands and -station- commands; a later -site set- command overrides an earlier one

```
site    set,'SITENAME'

Note:   zreturn = -1  if -site  set- is executed successfully
                = 0   if the lesson is not a site lesson for the
                      specified sitename
```

site	info	stores current site ECS information for the site specified by a preceding -site set-

```
site    info,STARTING VAR FOR STORING INFORMATION

Note:   Information consists of:

n(x)  or  nc(x) = starting variable
n(x)    contains the base ECS allotment
n(x+1)     "     the ECS currently allotted
n(x+2)     "     the ECS currently in use
n(x+3)     "     the number of active terminals at the site

        zreturn = -1  if -site  info- is executed successfully
                = 0   if no site has been set by -site  set-
```

site	active	stores the physical station numbers of the specified number of active stations on the site

```
site    active,STARTING STATION NUMBER,STARTING VAR FOR STORING
        STATION NUMBERS,NUM ACTIVE STATIONS TO STORE

Note:   zreturn = -1  if -site  active- is executed successfully
                = 0   if no site has been set by -site  set-
                = +1  if the starting station number is invalid
```

site stations stores the physical station numbers of the specified number of stations permanently on the site

```
site    stations,STARTING STATION NUMBER,STARTING VAR FOR STORING
        STATION NUMBERS,NUM STATIONS TO STORE
```

```
Note:  zreturn = -1  if -site  stations- is executed successfully
               =  0  if no site has been set by -site  set-
               = +1  if the starting station number is invalid
```

Station commands

These commands, all of which have -station- in the command field, give information on individual stations on the specified logical site.

station info stores information on the specified physical station number

```
station info,STATION NUMBER,STARTING VAR FOR STORING DATA
```

Note: Information consists of:

n(x) <u>or</u> nc(x) = starting variable

n(x) and n(x+1)	contain	the user's name (up to 18 characters)
n(x+2)	contains	the name of the user's group
n(x+3)	"	the type of user
n(x+4)	"	the account name containing the user's group
n(x+5)	"	session statistics (in 3 20-bit fields-- disk accesses, seconds of CPU, elapsed time in seconds)
n(x+6)	"	the name of the user's lesson or type of activity
n(x+7)	"	total ECS the lesson is using (in 3 15-bit fields-- storage ECS, common ECS, lesson ECS [left-most 15-bit field is empty])
n(x+8)	"	name of the user's router
n(x+9)	"	ECS used by the router (same format as lesson ECS)

zreturn = -1 if -station info- is executed successfully
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the station is inactive

station status sets "zreturn" according to the status of the specified physical station number

```
station status,STATION NUMBER
```

Note: zreturn = -2 if the station is in the process of signing on
= -1 if the station is active
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the station is inactive
= 4 if a backout of the station is in progress
= 5 if the station is locked out

station send sends the specified message (in -mode rewrite-) to the specified physical station number

station send,STATION NUMBER,SCREEN LOCATION,MESSAGE,NUM CHARACTERS IN MESSAGE

Note: zreturn = -1 if message is sent successfully
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 no message is sent (specified station is a runner station or is the station sending the message)

station logout backs out the station (given by the physical station number) (sets "backout" to -2)

station logout,STATION NUMBER

Note: zreturn = -1 if the backout is successful
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the specified station cannot be backed out

station stop1 presses STOP1 at the specified station (sets "backout" to +1 until the station enters another instructional lesson)

station stop1,STATION NUMBER

Note: zreturn = -1 if STOP1 is pressed at the station
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if STOP1 cannot be pressed at the specified station

station off turns off the specified station and prevents further use of the terminal (sets "backout" to -2)

station off,STATION NUMBER

Note: zreturn = -1 if the station is turned off successfully
= 0 if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the specified station cannot be turned off

station on turns on the specified station

`station on,STATION NUMBER`

```
Note:  zreturn = -1  if the station is turned on successfully
               =  0  if no site has been set by -site  set-
               =  1  if the starting station number is invalid
               =  2  if the specified station is not on the site
               =  3  the station is already active
```

Additional notes on MANAGING SITES

Basic display

at specifies starting position of display on the screen; sets left margin

```
at      COARSE
at      FINEX,FINEY
```

Note: The following formulas convert between character grid and fine grid.

```
finex = 800×frac(coarse/100) - 8
finey = 512 - 16×int(coarse/100)
coarse = 100×{1 + int[(511 - finey)/16]} + int(finex/8) + 1
```

atnm like -at- but does not reset the left margin

```
atnm    COARSE
atnm    FINEX,FINEY
```

write displays text on the screen

```
write   ANY MESSAGE, WHICH MAY CONSIST OF SEVERAL LINES
        AND INCLUDE EMBEDDED INFORMATION.
```

writec displays one of several messages depending on the rounded value of the conditional expression

```
writec  EXPR↕MESSAGEM↕MESSAGE0↕MESSAGE1↕↕MESSAGE3
```

NOTE: The embed feature is available. See descriptions of the individual commands in this section for information on FORMAT, MINIMUM, and ASTERISK, which are optional.

embedded -show-	⟨ show,EXPR,FORMAT,MINIMUM⟩ or ⟨ s,EXPR,FORMAT,MINIMUM⟩
embedded -showz-	⟨ showz,EXPR,FORMAT⟩ or ⟨ z,EXPR,FORMAT⟩
embedded -showt-	⟨ showt,EXPR,FORMAT⟩ or ⟨ t,EXPR,FORMAT⟩
embedded -showe-	⟨ showe,EXPR,FORMAT,ASTERISK⟩ or ⟨ e,EXPR,FORMAT,ASTERISK⟩
embedded -showo-	⟨ showo,EXPR,FORMAT⟩ or ⟨ o,EXPR,FORMAT⟩
embedded -showh-	⟨ showh,EXPR,FORMAT⟩ or ⟨ h,EXPR,FORMAT⟩
embedded -showa-	⟨ showa,STARTING VAR,FORMAT⟩ or ⟨ a,STARTING VAR,FORMAT⟩
embedded -at-	⟨ at,COARSE⟩ ; ⟨ at,FINEX,FINEY⟩
embedded -atnm-	⟨ atnm,COARSE⟩ ; ⟨ atnm,FINEX,FINEY⟩
embedded -size-	⟨ size,EXPR GIVING SIZE OF WRITING⟩ ⟨ size,SIZE IN X DIRECTION,SIZE IN Y DIRECTION⟩
embedded -rotate-	⟨ rotate,EXPR GIVING ANGLE FOR WRITING⟩
embedded -mode-	⟨ m,w⟩ (write mode) ⟨ m,e⟩ (erase mode) ⟨ m,r⟩ (rewrite mode)

NOTE: In the show-type commands (-show-, -showz-, -showt-, -showe-, -showo-, -showh-, -showa-), the general form is

```
showz   EXPR,FORMAT
```

where FORMAT, which may be an expression, is optional. If FORMAT equals 0, nothing is displayed.

For displaying entire arrays, the general form is

```
showt   ARRAYNAME,FORMAT   (for arrays with number of rows ≤16,
             number of columns ≤64; -showa-, -showh-, -showo-, -showe-
             may also be used to display entire arrays)
```

To display only the first element of an array, use

```
show    ARRAYNAME,FORMAT  or  showz   ARRAYNAME,FORMAT
```

show	displays the value of a variable or an expression with the specified number of significant digits but with suppression of trailing zeros after the decimal point; exponential format is displayed if the number of digits preceding the decimal point exceeds FORMAT by more than 4, or if the absolute value is less than 10^{-4}; MINIMUM is between 0 and 1 and specifies the smallest non-zero value to be displayed (0 is displayed if the absolute value of the expression is less than MINIMUM) show EXPR,NUM DIGITS,MINIMUM (FORMAT, if omitted, is 4; MINIMUM, if omitted, is 10^{-9})
showz	similar to -show- but displays all digits, including trailing zeros showz EXPR,NUM DIGITS (FORMAT, if omitted, is 4)
showt	displays the value of a variable or an expression in the specified format showt EXPR,NUM DIGITS PRECEDING DECIMAL POINT,NUM DIGITS FOLLOWING DECIMAL POINT (may be omitted if zero) (FORMAT, if omitted, is 4,3 for v-variable, 8 for n-variable; if the number of decimal places is less than 10, FORMAT may also be expressed as a single decimal number: e.g., 4.3 is equivalent to 4,3)
showe	displays the value of a variable or an expression in exponential format with the specified number of significant digits, including a leading blank or a minus sign; an optional third argument specifies the format for the exponent showe EXPR,NUM DIGITS,ASTERISK (FORMAT, if omitted, is 4; ASTERISK is omitted or =0 for exponent expressed by superscript, ≠0 for exponent expressed by 2 asterisks and multiplication sign replaced by one asterisk)

showo displays the value of a variable or an expression in octal notation

```
showo   EXPR,NUM DIGITS DISPLAYED   (FORMAT, if omitted, is 21;
           in embedded -showo- default format is 20)
```

showh displays the value of a variable or an expression in hexadecimal notation

```
showh   EXPR,NUM DIGITS DISPLAYED   (FORMAT, if omitted, is 16;
           in embedded -showh- default format is 15)
```

showa displays characters in the specified variable(s) or specified string

```
showa   STARTING VAR,NUM CHARACTERS   (FORMAT, if omitted, is 10)
showa   'STRING'   (STRING may contain up to 10 characters)
```

hidden displays hidden as well as visible characters; (special symbols are used to display hidden characters); number of characters includes all 6-bit character codes

```
hidden  STARTING VAR,NUM CHARACTERS (opt)   (NUM CHARACTERS,
                                             if omitted, is 10)
```

text displays contents of an alphanumeric buffer, line by line; the end of a line must be indicated by a variable ending with 2 zero codes (i.e., 12 bits equal to 0: o0000) (embedded zero codes (o00) are ignored); not affected by -size- or -rotate-

```
text    STARTING VAR,NUM VARS
```

erase erases the screen, selectively or entirely

```
erase   abort   (causes a full-screen erase and aborts output)
erase   (B) or  erase   NEGATIVE NUMBER   (causes full-screen erase
                                           but does not abort output)
erase   NUM CHARACTERS TO BE ERASED
erase   NUM CHARACTERS PER LINE,NUM LINES   (causes block erase)
```

Note: Selective erase is affected by preceding -size-, -gorigin- (and -scalex-, -scaley-), and -rorigin-.

lang sets the system variable "zlang"

```
lang    english   (sets "zlang" to 0)
lang    french    (sets "zlang" to 1)
lang    spanish   (sets "zlang" to 2)
lang    german    (sets "zlang" to 3)
lang    EXPR,french,x,english,german   (example of conditional form;
           argument x leaves "zlang" unchanged)
```

mode specifies terminal writing mode (see also system variable "mode")

```
mode    write      (normal writing state; writes only selected dots)
mode    erase      (erases only selected dots)
mode    rewrite    (erases and rewrites in one step; does not work
                   with "size" ≠ 0)
mode    inverse    (only on programmable terminal; displays dark
                   writing on light background)
mode    EXPR,erase,write,x,erase    (example of conditional form;
           argument x leaves writing mode unchanged)
```

Note: The mode is reset to "write" after any full-screen erase, in particular at a main unit not containing -inhibit erase-.

size specifies the size of line-drawn characters; remains in effect across main unit boundaries until turned off explicitly (see also system variables "size", "sizex", "sizey")

```
size    EXPR GIVING SIZE OF CHARACTERS
size    SIZE IN X DIRECTION,SIZE IN Y DIRECTION    (sets independent
           sizes in x and y directions)
size    0  or  size    (B)    (restores standard writing)
```

Note: Negative "sizex" gives backwards characters and writing; negative "sizey" gives upside down characters and writing. Negative "size" behaves like simultaneous negative "sizex" and negative "sizey".

rotate causes line-drawn characters to be written at the angle specified in the tag; remains in effect across main unit boundaries until turned off explicitly (must be used with -size- tag ≠ 0)

```
rotate  EXPR GIVING ANGLE IN DEGREES (omit degree symbol;
        measured counter-clockwise from horizontal)
rotate  0  or  rotate  (B)    (restores horizontal writing)
```

delay permits short delays during output

```
delay   DURATION OF OUTPUT DELAY IN FRACTIONS OF A SECOND, UP TO
        1 SECOND (accurate to 1/60 second)
```

Note: This command causes "do-nothing" output to be sent to the terminal for the specified delay time.

inhibit temporarily disables certain normal actions of TUTOR in a unit; all settings are cleared at each main unit

inhibit anserase	(prevents automatic erasure of answer-contingent message when a response is erased)
inhibit arrow	(prevents plotting of the response arrow)
inhibit blanks	(prevents judging if NEXT is pressed before any characters are typed)
inhibit charclear	(prevents clearing of the charset flag)
inhibit dropfile	(prevents the attached file from being released during a jumpout)
inhibit dropstor	(prevents storage from being dropped during a jumpout)
inhibit edit	(prevents use of the EDIT key)
inhibit erase	(prevents normal full-screen erase when proceeding to the next main unit)
inhibit from	(prevents return to the lesson containing this statement via -jumpout return- or -jumpout return,return-)
inhibit jumpchk	(prevents ECS check before attempting a jumpout)
inhibit term	(prevents use of the TERM key)
inhibit (B) <u>or</u> inhibit clear	(removes effect of previous -inhibit- commands in that main unit)
inhibit clear,arrow,blanks	(may combine tags)

char permits specification of specially designed characters for display

```
char    NAME,ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7,ARG8
```

Note: The character name (NAME) may be a number from 0 to 126 (excluding 63) or a defined name. Arguments ARG1 through ARG8 are numbers which specify which of the 16 dots are lit in each of the 8 columns of the character space.

for example:

```
define  chi=88    $$ load chi on X
char    chi,o4020,o10040,o6300,o1600,o3140,o4020,o10040,0
```

plot displays a special character previously specified by a -char- command

```
plot    NAME
plot    EXPR   (EXPR may have value from 0 to 126, excluding 63)
```

Note: Special characters may also be displayed by pressing the FONT key and then the key(s) where the character(s) are loaded into the terminal memory. Built-in characters are displayed after FONT is pressed again.

Graphics

NOTE: With -dot-, -draw-, -box-, -fill-, -vector-, -window-, LOCATION is the screen location and may be COARSE or FINEX,FINEY. Coarse grid and fine grid coordinates may be mixed in tags with more than one argument.

dot draws a dot at the specified screen location

```
dot     LOCATION
```

draw draws a dot, line, or line-drawn figure; after execution, "wherex" and "wherey" are set to the last point plotted

```
draw    LOCATION   (draws a dot; -dot- is faster if many dots are
            plotted; -draw- is faster if lines are also being drawn)
draw    LOCATION1;LOCATION2   (draws a line)
draw    LOCATION1;LOCATION2;LOCATION3;...   (draws connected lines)
draw    ;LOCATION   (draws a continued line)
draw    LOCATION1;LOCATION2;skip;LOCATION3;LOCATION4
           ("skip" moves to a new position without plotting)
```

Note: Maximum number of numbers in the tag is 63 ("skip" counts as a number).

box draws a rectangle with the specified corner locations and thickness; after execution, "wherex", "wherey" are set to the lower left corner of the box with thickness included

```
box     CORNER LOCATION;OPPOSITE CORNER LOCATION;DOTS THICK(opt)
box     ;CORNER LOCATION;DOTS THICK (opt)  (opposite corner at current
                                            "wherex", "wherey")
box     CORNER LOCATION    (opposite corner at 0,0; cannot specify
                            thickness with this form of tag)
box     (B)   (equivalent to  -box  0,0;511,511-)
```

Note: Thickness, if omitted, 0, 1, or -1, is 1 dot. Negative thickness extends inward; positive thickness extends outward. Maximum thickness is 95 (or -95).

fill fills in a rectangular area on the screen on programmable terminals; does not affect the setting of "wherex", "wherey"

```
fill    CORNER LOCATION;OPPOSITE CORNER LOCATION
fill    ;CORNER LOCATION   (opposite corner at "wherex", "wherey")
fill    (B)   (fills in the entire screen, i.e., corners 0,0;511,511)
```

vector draws a vector symbol with specified tail and head locations and head size

```
vector  TAIL LOCATION;HEAD LOCATION;SIZE (opt)
vector  ;HEAD LOCATION;SIZE (opt)  (tail at "wherex", "wherey")
vector  HEAD LOCATION   (tail at 0,0 ; cannot specify head size)
vector  0,0;HEAD LOCATION;SIZE   (tail at 0,0 ; head size specified)
```

Note: SIZE, if omitted, is 10 or 11 dots for moderate-length vectors. Negative size indicates open arrowhead. |size|≥1 is absolute (in screen dots); |size|<1 is relative to the length of the vector.

window establishes a rectangular "window" on the screen outside of which line-drawn display is not plotted; remains in effect across main unit boundaries until turned off explicitly

```
window  CORNER LOCATION;OPPOSITE CORNER LOCATION
window  ;CORNER LOCATION  (opposite corner at "wherex", "wherey")
window  CORNER LOCATION   (opposite corner at 0,0)
window  (B)   (clears previous -window- setting)
```

circle draws a circle with the specified parameters; the center is at the current "wherex", "wherey"; after execution, "wherex", "wherey" are set to the center for a one-argument tag and to the end of the last line drawn for the three-argument tag

```
circle  RADIUS IN DOTS,START ANGLE (opt),END ANGLE (opt)
            (second and third arguments are optional: if omitted,
            START ANGLE is 0° and END ANGLE is 360°; angles
            are measured in degrees counter-clockwise from
            START ANGLE; degree sign is omitted)
```

circleb same options as -circle- but draws a broken circle

```
circleb RADIUS IN DOTS,START ANGLE (opt),END ANGLE (opt)
```

Relocatable graphics

rorigin establishes a "relocatable" origin for use with -rdraw-, -rat-, -rbox-, -rvector-, and -rcircle-

```
rorigin COARSE
rorigin FINEX,FINEY
rorigin (B)   (sets relocatable origin to "wherex", "wherey")
```

Note: Upon entering a lesson, the relocatable origin is automatically set to -rorigin 0,0-.

NOTE: All subsequent relocatable commands are affected by preceding -rorigin-, -size-, and -rotate-.

rat similar to -at- but relative to the -rorigin- location; affected by preceding -size- and -rotate-

```
rat     X-LOCATION,Y-LOCATION
rat     (B)   (equivalent to -rat 0,0-, i.e., the current
              -rorigin- location)
```

ratnm similar to -rat- but does not reset the left margin (see -atnm-)

```
ratnm   X-LOCATION,Y-LOCATION
```

rdot draws a dot at the specified position relative to the -rorigin- location; position is affected by preceding -size- and -rotate-

```
rdot    X-LOCATION,Y-LOCATION
```

rdraw similar to -draw- but figure is affected by preceding -size- and/or -rotate-; the last point plotted serves as the location for the next screen activity

```
rdraw   TAG LIKE -draw- EXCEPT WITH RESPECT TO -rorigin- LOCATION
        (i.e., in screen dots from the -rorigin- location)
```

rbox similar to -box- but draws a rectangle relative to the -rorigin- location; affected by preceding -size- and -rotate- (see -box-)

```
rbox    CORNER X,CORNER Y;OPPOSITE CORNER X,OPPOSITE CORNER Y;
        DOTS THICK (opt)
rbox    ;CORNER X,CORNER Y;DOTS THICK (opt)   (opposite corner at
                                               "wherex", "wherey")
rbox    CORNER X,CORNER Y   (opposite corner at -rorigin- location;
           cannot specify thickness with this form of tag)
```

rvector	similar to -vector- but draws vector symbol relative to the -rorigin- location; affected by preceding -size- and -rotate- (see -vector-) rvector XTAIL,YTAIL;XHEAD,YHEAD;SIZE (opt) rvector ;XTAIL,XHEAD;SIZE (opt) (tail at "wherex", "wherey") rvector XHEAD,YHEAD (tail at -rorigin- location) rvector 0,0;XHEAD,YHEAD;SIZE (tail at -rorigin- location)
rcircle	same options as -circle- but is affected by preceding -rotate- and -size-; gives an ellipse if preceded by two-argument -size- with unequal arguments (see -circle-) rcircle RADIUS IN DOTS,START ANGLE (opt),END ANGLE (opt) (specify basic radius before affected by -size-)

Drawing graphs

gorigin specifies location of the origin of the graph; all other display with graphing commands is relative to this origin

```
gorigin COARSE
gorigin FINEX,FINEY
gorigin (B)   (sets graph origin to "wherex", "wherey")
```

Note: Upon entering a lesson, the origin is automatically set to -gorigin 0,0-.

axes specifies lengths of the axes and draws the axes; remains in effect across main unit boundaries until reset; x and y values are in dots relative to the -gorigin- location

```
axes    NEGATIVE X,NEGATIVE Y,POSITIVE X,POSITIVE Y
axes    POSITIVE X,POSITIVE Y
axes    (B)   (draws axes specified by the last -axes- or -bounds-)
```

Note: To draw one-quadrant axes (other than both positive axes) with labeling on the outside of the axes, use four-argument form of the tag with arguments corresponding to missing axes set to 0.

bounds specifies lengths of the axes but does not draw the axes (i.e., axes are invisible); remains in effect across main unit boundaries until reset; x and y values are in dots relative to the -gorigin- location

```
bounds  NEGATIVE X,NEGATIVE Y,POSITIVE X,POSITIVE Y
bounds  POSITIVE X,POSITIVE Y
bounds  (B)   (sets up bounds specified by the last -axes- or -bounds-)
```

Note: Upon entering a lesson the boundaries are automatically set to -bounds 511,511-.

scalex specifies the maximum value and the value at the origin on the x axis; remains in effect across main unit boundaries until reset

```
scalex  MAXIMUM VALUE OF X,VALUE OF X AT ORIGIN (opt)
             (value at origin, if omitted, is 0)
```

scaley same options as -scalex- but for the y axis

```
scaley  MAXIMUM VALUE OF Y,VALUE OF Y AT ORIGIN (opt)
             (value at origin, if omitted, is 0)
```

lscalex specifies the maximum value and the value at the origin on the x axis; the scale between these points is proportional to the logarithm of maximum x divided by the value at the origin; remains in effect across main unit boundaries until reset

lscalex MAXIMUM VALUE OF X,VALUE OF X AT ORIGIN (opt)
(value at origin, if omitted, is 1, i.e., 10^0)

lscaley same options as -lscalex- but for the y axis

lscaley MAXIMUM VALUE OF Y,VALUE OF Y AT ORIGIN (opt)
(value at origin, if omitted, is 1, i.e., 10^0)

NOTE: If any of the commands -scalex-, -scaley-, -lscalex-, -lscaley- are omitted, a linear scale with length set by the preceding -axes- or -bounds- is assumed.

NOTE: Subsequent graphing commands are in appropriate scaled units.

labelx draws tick marks and labels the x axis

labelx MAJOR INTERVAL,MINOR INTERVAL(opt),MARKSIZE(opt),FORMAT(opt)

labely draws tick marks and labels the y axis

labely MAJOR INTERVAL,MINOR INTERVAL(opt),MARKSIZE(opt),FORMAT(opt)

markx draws tick marks on the x axis with no labels

markx MAJOR INTERVAL,MINOR INTERVAL(opt),MARKSIZE(opt),FORMAT(opt)

marky draws tick marks on the y axis with no labels

marky MAJOR INTERVAL,MINOR INTERVAL(opt),MARKSIZE(opt),FORMAT(opt)

NOTE: Commands -labelx-, -labely-, -markx-, -marky- specify major and minor mark intervals.
MARKSIZE = 0 or omitted for normal label marks
MARKSIZE = 1 for major marks extending to bounds of the graph
MARKSIZE = 2 for all marks extending to bounds of the graph
MINOR INTERVAL may be omitted. If MAJOR INTERVAL is set to 0 with linear scale, the computer chooses the "best" interval.
The total number of marks on an axis cannot exceed 100.
FORMAT gives the format for the labels and has the same form as that for -showt-, e.g., 1.2 or 1,2. FORMAT is optional; if omitted, the label format is selected automatically.

(NOTE continued on next page.)

NOTE: (continued from preceding page)
For labeling log scales:
MAJOR INTERVAL must be 0 (major marks are automatically plotted every decade)
MINOR INTERVAL < 0, minor marks are not plotted
MINOR INTERVAL = 0 or 3 (or omitted), minor marks are placed at values of 1, 2, 5 within the decade
MINOR INTERVAL = 5, minor marks are placed at 1, 2, 3, 5, 7
MINOR INTERVAL = 10, minor marks are placed at 1, 2, 3, 4, 5, 6, 7, 8, 9

polar causes tags of graphing commands to be interpreted as polar coordinates containing scaled radius and polar angle; may set scales on x and y axes; remains in effect across main unit boundaries until turned off explicitly; polar conversion and scaling must be turned off independently

```
polar   (B)   (turns on polar conversion)
polar   MAXIMUM VALUE OF X AND Y   (turns on polar conversion
           and scales both axes)
polar   MAXIMUM VALUE OF X,MAXIMUM VALUE OF Y   (turns on
           polar conversion and scales axes independently)
polar   NEGATIVE VALUE   (turns off polar conversion but not scale)
```

NOTE: When the tag of subsequent commands is interpreted in polar coordinates, the degree sign must be used if the angle is in degrees. Without the degree sign, the angle is interpreted in radians.

gat similar to -at- but specifies the screen location relative to the -gorigin- location and in scaled units

```
gat     X-LOCATION,Y-LOCATION
gat     DISTANCE,ANGLE   (with -polar-)
gat     (B)   (equivalent to -gat  0,0-, i.e., the current
              -gorigin- location)
```

gatnm similar to -gat- but does not reset the left margin (see -atnm-)

```
gatnm   X-LOCATION,Y-LOCATION
gatnm   DISTANCE,ANGLE   (with -polar-)
```

gdot draws a dot at the specified position relative to the -gorigin- location and in scaled units

```
gdot    X-LOCATION,Y-LOCATION
gdot    DISTANCE,ANGLE   (with -polar-)
```

graph places a dot or character string centered at the position indicated relative to the -gorigin- location and in scaled units

```
graph   X-LOCATION,Y-LOCATION,STRING (opt)   (maximum of 9
           characters in string; if string is not specified, a
           dot is plotted)
graph   DISTANCE,ANGLE,STRING   (with -polar-)
graph   X-LOCATION,Y-LOCATION;VAR,NUM CHARACTERS (opt)
           (if number of characters is omitted, 10 characters
           are plotted)
graph   DISTANCE,ANGLE;VAR,NUM CHARACTERS   (with -polar-)
```

gdraw like -draw- but relative to the -gorigin- location and in scaled units; after execution "wherex", "wherey" are set to the last point plotted

for example:

```
gdraw   X1,Y1;X2,Y2   (draws a line on the graph)
gdraw   DISTANCE1,ANGLE1;DISTANCE2,ANGLE2   (with -polar-)
```

gbox same options as -box- but draws a rectangle relative to the -gorigin- location; affected by preceding -scalex- and -scaley- (see -box-)

```
gbox    CORNER X,CORNER Y;OPPOSITE CORNER X,OPPOSITE CORNER Y;
        DOTS THICK (opt)
gbox    DISTANCE CORNER,ANGLE CORNER;DISTANCE OPPOSITE CORNER,
        ANGLE OPPOSITE CORNER;DOTS THICK   (with -polar-)
gbox    ;CORNER X,CORNER Y;DOTS THICK (opt)   (draws a box
           with opposite corner at current "wherex", "wherey")
gbox    ;DISTANCE CORNER,ANGLE CORNER;DOTS THICK   (with -polar-)
gbox    CORNER X,CORNER Y   (draws a box with opposite corner at
           -gorigin- location; cannot specify thickness with this
           form of tag)
gbox    DISTANCE CORNER,ANGLE CORNER   (with -polar-)
gbox    (B)   (draws a box set by a previous -axes-/-bounds- and
              -scalex-/-scaley-)
```

gcircle same options as -circle- but is affected by preceding -scalex- and -scaley-; draws an ellipse if the -scalex- and -scaley- settings are different (see -circle-)

```
gcircle RADIUS IN DOTS,START ANGLE (opt),END ANGLE (opt)
           (specify basic radius before affected by -scalex-, -scaley-)
```

gvector same options as -vector- except draws vector symbol relative to the -gorigin- location and in scaled units (see -vector-)

```
gvector XTAIL,YTAIL;XHEAD,YHEAD;SIZE (opt)
gvector DISTANCETAIL,ANGLETAIL;DISTANCEHEAD,ANGLEHEAD;SIZE(opt)
            (with -polar-)
gvector ;XHEAD,YHEAD;SIZE (opt)   (tail at "wherex", "wherey")
gvector ;DISTANCE HEAD,ANGLE HEAD;SIZE (opt)   (with -polar-)
gvector XHEAD,YHEAD   (tail at -gorigin- location)
gvector 0,0;XHEAD,YHEAD;SIZE   (tail at -gorigin- location)
gvector LENGTH,ANGLE   (tail at -gorigin- location; with -polar-)
```

Note: Because of the default conditions of -gorigin 0,0- and -bounds 511,511-, -gvector- used without preceding -gorigin- and -bounds- gives the same result as -vector- with fine-grid coordinates.

vbar draws a vertical bar at the specified location relative to the -gorigin- location and in scaled units

```
vbar    X-LOCATION,HEIGHT,STRING (opt)
vbar    DISTANCE BAR TOP,ANGLE BAR TOP,STRING   (with -polar-)
vbar    X-LOCATION,HEIGHT;VAR,NUM CHARACTERS (opt)
vbar    DISTANCE BAR TOP,ANGLE BAR TOP;VAR,NUM CHARACTERS
            (with -polar-)
```

hbar draws a horizontal bar at the specified location relative to the -gorigin- location and in scaled units

```
hbar    LENGTH,Y-LOCATION,STRING (opt)
hbar    DISTANCE BAR END,ANGLE BAR END,STRING   (with -polar-)
hbar    LENGTH,Y-LOCATION;VAR,NUM CHARACTERS (opt)
hbar    DISTANCE BAR END,ANGLE BAR END;VAR,NUM CHARACTERS
            (with -polar-)
```

NOTE: With -vbar- and -hbar-, STRING may have up to 9 characters. If STRING is omitted, a rectangle is drawn. If the character string is stored in a variable and number of characters is omitted, 10 characters are drawn.

delta specifies stepsize for subsequent -funct- commands

delta STEPSIZE

Note: If -delta- is omitted, the stepsize is set to 1.

funct plots the curve specified in the tag, with the stepsize given by a preceding -delta- or by the stepsize given in the tag

funct FUNCTION EXPR,INDEPENDENT VAR

Note: Range of independent variable is set by boundaries of -axes- (or -bounds-) and -scalex- commands.

funct FUNCTION EXPR,INDEPENDENT VAR← INITIAL,FINAL,STEPSIZE

Note: If initial or final values of the independent variable are beyond previously set boundaries, the latter are used. For polar functions if initial or final value is omitted, it is assumed to be 0 or 2π, respectively.

With either form of -funct-, a v-variable is recommended for the independent variable.

NOTE: With -delta- and -funct-, select a stepsize that gives a smooth graph but plots quickly. A reasonable lower limit to the stepsize for a graph with linear x axis is:

$|\text{STEPSIZE}| \geq .02 \times |\text{FINAL VALUE} - \text{INITIAL VALUE}|$.

Non-screen

slide operates the slide projector and selects the specified slide

```
slide   SLIDE NUMBER   (value from 0 to 255)
slide   ROW+16×COLUMN   (ROW, COLUMN from 0 to 15)
slide   512   (turns off lamp)
slide   256   (closes shutter)
slide   512+SLIDE NUMBER   (selects slide with lamp off)
slide   256+SLIDE NUMBER   (selects slide with shutter closed)
slide   noslide   (selects slide 0, turns off lamp, closes shutter)
```

audio sends the value of the tag (truncated to 15 bits) to the external device connected to the "audio" jack

```
audio   EXPR
```

play plays the audio device recording at the message location specified

```
play    TRACK,SECTOR,NUM SECTORS   (128 tracks, 32 sectors each)
```

record records a message at the location specified

```
record  TRACK,SECTOR,NUM SECTORS
```

enable allows input from the touch panel and from external devices

```
enable  touch
enable  ext
enable  touch,ext   (may combine tags)
```

Note: -enable touch- must be reset for each -arrow- command in a unit and after any full-screen erase.
-enable touch- in a unit with no -arrow- allows any touch on the screen to have the effect of pressing NEXT.
-enable ext- is turned off only by -disable ext-.

disable prevents input from any device except the keyset; this is the normal state of the terminal

```
disable touch
disable ext
disable touch,ext
```

ext sends the value of the tag (truncated to 15 bits) to an external device (or to the device at another station if "ext" option has been turned on by the receiving user)

```
ext     EXPR
ext     EXPR,STATION

Note:  zreturn = -1 if data is sent successfully
               =  0 otherwise
```

extout sends the value of the right-most 16 bits of the specified variables to an external device; the 16th bit from the right determines how the information is interpreted: 1 for ext, 0 for audio

```
extout  STARTING VAR,NUM VARS (opt)   (NUM VARS, if omitted, is 1)
```

xout sends data (in 8-bit bytes) contained in the specified variables to an external device (available only on programmable terminals)

```
xout    DEVICE ADDRESS   (establishes an address for use by subsequent
                         -extout- or -ext- commands)
xout    ADDRESS,STARTING VAR,NUM BYTES,SEGMENT SIZE (opt)
           (SEGMENT SIZE, if omitted, is 60; if SEGMENT SIZE > 8,
           only the right-most 8 bits are sent)

Note:  zreturn = -1  if the data is sent successfully
               =  0  if STOP or STOP1 is pressed during transmission
```

xin collects data (in 8-bit bytes) from an external device and stores it in the specified variables (available only on programmable terminals)

```
xin     DEVICE ADDRESS   (establishes an address to be read upon
                         subsequent external interrupt requests)
xin     ADDRESS,STARTING VAR,NUM BYTES,SEGMENT SIZE (opt)
           (SEGMENT SIZE, if omitted, is 60; if SEGMENT SIZE > 8,
           the right-most 8 bits of "key" are stored, right-justified,
           in each segment)

Note:  zreturn = -1  if the data is received successfully
               =  0  if STOP or STOP1 is pressed during transmission
```

beep (no tag) rings the sound device on programmable terminals

NOTE: Commands -xout-, -xin-, -beep- may be used only at a programmable terminal.
For current information on device addresses, see the descriptions of -xin- or -xout- in "aids".

saylang specifies the language to be spoken by a phonemic synthesizer which is operated by the terminal (languages currently available: WES [World English Spelling], ipa [International Phonetic Alphabet], Esperanto, and Spanish); currently works only with Votrax model VS-6

```
saylang LANGUAGE
saylang (B)  or  saylang  q   (turns off subsequent -say- commands)
saylang EXPR,LANGUAGEM,LANGUAGE0,q,LANGUAGE2,x   (example of
             conditional form)
```

say specifies the sentence to be spoken by the synthesizer

```
say     SENTENCE OR PHRASE   (may include embedded information)
```

sayc specifies the sentence to be spoken by the synthesizer depending on the value of a conditional expression

```
sayc       EXPR↕PHRASEM↕PHRASE0↕PHRASE1 ...
```

Special display

tabset sets tabs which are used by a student pressing the TAB key

```
tabset  OCTAL NUMBER CONTAINING 10 PACKED TAB SETTINGS FROM
        LEFT TO RIGHT  (each setting is a 6-bit octal number giving
           the horizontal character position; unused settings to
           the right must be filled with octal zeros)
```

for example, to set tabs at horizontal character positions 8, 21, 30, 48, 56, and 63, use:

```
     tabset  o10 25 36 60 70 77 00 00 00 00
```

Note: The tag may be an n-variable which contains the packed settings.

micro specifies microtable definitions used when the student presses the MICRO key and then another key; unless specified, microtable and -micro- command are in the same lesson

```
micro    (0),MICROTABLE NAME
micro    (zlesson),MICROTABLE NAME
micro    ,MICROTABLE NAME
micro    MICROTABLE NAME
micro    LESSON NAME,MICROTABLE NAME  (LESSON NAME contains the
            microtable; enclose variable arguments in parentheses)
micro    <LESLIST POSITION>,MICROTABLE NAME
micro    (B)   (cancels microtable in effect and restores built-in
               microtable definitions)
```

```
Note:  zreturn = -1  if the microtable is available
               =  0  if the microtable is not found
```

charset causes the specified character set to be loaded into the terminal memory (see -inhibit charclear-); 1-block charset may contain up to 79 characters, 2-block charset up to 126 characters; unless specified, the charset and -charset- command are in the same lesson

```
charset (0),CHARSET NAME
charset (zlesson),CHARSET NAME
charset ,CHARSET NAME
charset CHARSET NAME
charset LESSON NAME,CHARSET NAME  (LESSON NAME contains the charset
           blocks; enclose variable arguments in parentheses)
charset <LESLIST POSITION>,CHARSET NAME
charset (B)   (clears charset flag)
```

```
Note:  zreturn = -1  if the character set is loaded successfully
               =  0  if the character set is not found
               =  1  if the STOP key is pressed during loading
               =  2  if there is an error in loading
               =  3  if there is a disk error
               =  5  if the variable for the charset name equals 0
```

chartst allows check for presence of a character set in the terminal memory; sets "zreturn" to -1 if the charset flag is set and to 0 if not

```
chartst (0),CHARSET NAME
chartst (zlesson),CHARSET NAME
chartst ,CHARSET NAME
chartst CHARSET NAME
chartst LESSON NAME,CHARSET NAME  (LESSON NAME contains the
          charset; enclose variable arguments in parentheses)
chartst <LESLIST POSITION>,CHARSET NAME
```

lineset allows use of line-drawn characters, which are affected by preceding -size- and -rotate-; "size" must not be 0; linechars are accessed by the FONT key or by -altfont on- ; a lineset may be up to 3 blocks long; 1 block may contain up to 128 small linechars; unless specified, lineset blocks and -lineset- command are in the same lesson

```
lineset (0),LINESET NAME
lineset (zlesson),LINESET NAME
lineset ,LINESET NAME
lineset LINESET NAME
lineset LESSON NAME,LINESET NAME   (LESSON NAME contains the
          lineset; enclose variable arguments in parentheses)
lineset <LESLIST POSITION>,LINESET NAME
lineset (B)   (cancels lineset in effect and restores standard
               sized writing)
```

```
Note:  zreturn = -1  if the lineset is attached successfully
               =  0  if the lineset is not found
               = +1  if there is an error in the lineset
```

altfont changes font mode of the terminal; affects charsets and linesets

```
altfont on   or   altfont 1   or   altfont alt   (switches terminal
                                                  to alternate font)
altfont off   or   altfont 0   or   altfont normal   (switches
          terminal to normal font, which is the default state)
```

Note: Tag may be calculated, but it must be exactly 0 or 1. Altfont setting remains in effect across unit boundaries until reset by another -altfont- command or until -jumpout- is executed.

System variables for presenting

mode	= -1 with -mode erase- = 0 with -mode rewrite- = 1 with -mode write- = 2 with -mode inverse-	see -mode- command
size	current value of the tag of the single-argument -size- command (see -size- command)	
sizex	current value of the "x" argument in the two-argument -size- command	
sizey	current value of the "y" argument in the two-argument -size- command	
where	character-grid location for next display	See CALCULATING, System functions, for zfinex(X), zfiney(X)
wherex	fine-grid x location for next display	
wherey	fine-grid y location for next display	
zlang	useful for display of multi-lingual text; set by -lang- command = 0 for -lang english- = 1 for -lang french- = 2 for -lang spanish- = 3 for -lang german-	

Additional notes on PRESENTING

Router lesson

route used in a router lesson to specify to which unit in the router lesson the user is sent upon leaving the instructional lesson

```
route   end lesson,UNIT NAME  (exit via -end  lesson- or -jumpout  q-)
route   error,UNIT NAME   (exit via execution error or condense error)
route   finish,UNIT NAME   (exit via STOP1)
route   resignon,UNIT NAME (opt)   (upon STOP1 exit from a lesson,
           provides the user with a choice page offering the option
           to sign off completely or to continue working
           [i.e., to return to the router, to the specified unit,
           if given, or to the first unit if UNIT NAME is omitted])
```

routvar (non-executable) sets up special variables in a router lesson which can be used only in the router lesson; they are referenced by vr and nr

```
routvar NUM VARS   (maximum of 64 variables)
```

allow used in a router lesson to specify that router common and/or router variables may be referenced in the instructional lesson

```
allow   read    (read-only access to router common)
allow   write   (read and write access to router common)
allow   read rvars   (read-only access to router variables)
allow   (B)   (clears last setting of -allow-)
```

Curriculum information

NOTE: The following commands are used in instructional lessons.

lesson   sets the system variable "ldone" to indicate whether a lesson is considered complete

```
lesson  complete     (sets "ldone" to -1)
lesson  incomplete   (sets "ldone" to 0)
lesson  no end   (sets "ldone" to +1; may be used in lessons with
                 no logical end)
lesson  EXPR,complete,incomplete,x,no end   (example of conditional
           form; argument x leaves "ldone" unchanged)
```

score   places value of the tag, rounded to the nearest integer, into the system variable "lscore"

```
score   EXPR   (value from 0 to 100)
score   (B)    (sets "lscore" to -1)
```

status   places the value of the tag, rounded to the nearest integer, into the system variable "lstatus"; allows a student to reestablish a status (to some extent) upon returning to a lesson after having entered other lessons

```
status  EXPR
```

System variables for routing

errtype	= 0 for unknown error type = 1 for execution error = 2 for fatal condense error or for attempted jumpout to a router not specified for the group = 3 for memory exceeded = 4 for error in the finish unit of the instructional lesson = 5 for exit from the condense queue via STOP1
ldone	= -1 if the user has encountered -lesson complete- or -end lesson- = 0 if the user has encountered -lesson incomplete- or has never entered the lesson or has entered but not completed the lesson (-records ldonelist- returns a value of 2 for the last case when "mrouter" is used) = +1 if the user has encountered -lesson no end-
lscore	rounded value of the tag of -score- (value from 0 to 100); initially set to -1 for a student routed by "mrouter"; may also be set to -1 with -score (B)-; initially set to 0 for a student not routed by "mrouter"
lstatus	rounded value of the tag of -status-
rcallow	= 0 for no access to router common = 1 for -allow read- = 2 for -allow write-
router	name of the router lesson (left-justified; display with -showa-)
rstartl	name of the lesson from the last -restart- command (left-justified; display with -showa-)
rstartu	name of the unit from the last -restart- command (left-justified; display with -showa-)
rvallow	= 0 for no access to router variables = 1 for -allow read rvars-

zleserr gives detailed information on fatal errors which can occur when accessing a lesson (i.e., errors that give a student the message "Call Your Instructor")

= 0 if there is no error or if the error is non-fatal
= 1 if the condensor is not available
= 2 if the lesson does not exist
= 3 if the lesson source code is too long
= 4 if ECS is not available (although the site ECS allocation is not exceeded)
= 5 (system error)
= 6 if there is a disk error
= 7 if there is a unit which is too long
= 8 if the lesson has been deleted
= 9 (not used)
= 10 if there is no room in ECS for the lesson common
= 11 if the common is not found
= 12 if there are not enough common blocks
= 13 (system error)
= 14 if there is a common codeword error
= 15 if there is a tag which is too long
= 16 if the lesson binary is too long
= 17 if the lesson is not a TUTOR lesson
= 18 if the lesson is temporarily unavailable
= 19 if the site ECS allocation is exceeded
= 20 (system error)
= 21 (system error)
= 22 if there is an error in specifying the router
= 23 if there is a jumpout codeword error
= 24 if the common in ECS has a different length from the length specified in the -common- command
= 25 if a jumpout to the wrong router is attempted
= 26 if there is an error in the -use- command (other than "block not found")
= 27 (system error)
= 28 if the lesson is in the improper lesson access class (not used on the CERL system)
= 29 (not used)
= 30 if the lesson is obsolete and must be converted
= 31 if there is an error in the -use- command: block not found
= 32 if the processor lesson is not a valid TUTOR file

Basic sequencing

unit names and initiates a section of a lesson (called a unit) which may be referenced by other sequencing commands

```
unit    NAME   (maximum of 8 characters in NAME)
```

unitop similar to -unit- but without a full-screen erase when the unit is entered (except upon initial entry into a lesson); "mode" and "where" are not altered

```
unitop  NAME   (maximum of 8 characters in NAME)
```

NOTE: Initial entry unit (ieu) refers to commands preceding the first -unit- or -unitop- command in a lesson; these are executed whenever and wherever a lesson is entered (except when a lesson executes -jumpout- to itself or when a router lesson is returned to during the session).

See -define- for formats for a local define set, which is declared as a continuation of a -unit- command.

entry names a section of a lesson which may be referenced by other sequencing commands; does not affect the flow of execution of the unit in which the -entry- command is placed except that -entry- may not be placed within the range of -branch-, -doto-, -if-, or -loop-; no keypress is required to execute commands following -entry-; no full-screen erase or other main-unit initializations occur following -entry- when it is executed within a unit

```
entry   NAME   (maximum of 8 characters in NAME)
```

NOTE: Commands -unit-, -unitop-, and -entry- may have a form with arguments:

```
unit    NAME(VAR1,VAR2,VAR3,...)    (up to 10 arguments)
```

A lesson may have up to 394 different units referenced by -unit-, -unitop-, and -entry-. No unit may be named "q" or "x". Maximum length of a unit is 500 condensed words.

Automatic sequencing

NOTE: The following commands (-jump-, -goto-, -do-, and -join-) may have a conditional form, e.g.,

```
goto    EXPR,NAMEM,NAME0,NAME1,x,NAME3,q
do      EXPR,NAMEM,NAME0,x,NAME2,q,VAR⇐ INITIAL,FINAL,STEP
```

Argument x is equivalent to absence of the command; argument q is equivalent to a branch to an empty unit. Special case occurs with -do q- (-join q-), which is equivalent to -goto q-. For iterative -do-, q terminates the -do-, and x indicates no iteration is done for that value of the conditional expression. Argument q is not valid with -jump-. Up to 100 arguments are permitted in the conditional tag.

These commands may pass up to 10 arguments, e.g.,

```
goto    NAME(VALUE1,VALUE2,,VALUE4)  (values may be expressions)
  .
  .
  .
unit    NAME(VAR1,VAR2,VAR3,VAR4)   (VAR3 is unchanged)
```

<u>or</u>

```
do      NAME(VALUE1,VALUE2;VAR1,VAR2,VAR3)   (-return- returns
           values to VAR1, VAR1, VAR3)
```

jump causes execution of the unit named in the tag with a full-screen erase (unless the erase is prevented: see -inhibit erase-) and change of main unit; initializations associated with entering a main unit are performed

```
jump    UNIT NAME
```

goto causes execution of the unit named in the tag without a screen erase, without change of main unit, and without other main-unit initializations; there is no further execution of commands in the original unit except during the judging process

```
goto    UNIT NAME
```

do (insertion) causes execution of the unit named in the tag without screen erase or change of main unit; returns to the original unit to execute commands following -do-

(iteration) causes repeated execution of unit(s) named in the tag while changing a counter; otherwise same as insertion -do-

```
do      UNIT NAME
do      NAME,VAR⇐ INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
           (STEPSIZE, if omitted, is +1; STEPSIZE may be negative;
           loop variable is undefined after completion of the loop)
```

Note: Nested -do- and -join- levels may be up to 10 deep.

join similar to -do- but is executed during judging and during search for additional -arrow- commands following an "ok" judgment

```
join    UNIT NAME
join    NAME,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
```

return returns values to variables specified in a -unit- command with arguments

```
return  EXPR1,EXPR2,EXPR3   (maximum of 10 arguments)
```

Note: -return- occurs in a unit executed via -do- or -join-.

```
do      NAME(EXPR;VAR1,VAR2)
:
:
unit    NAME(VAR)
:
:
return  EXPR1,EXPR2   (returns values to VAR1, VAR2)
```

exit permits termination of -do- or -join- sequences

```
exit    (B)  or   exit    NEGATIVE VALUE    (exit from all levels
                                             of -join- and -do-)
exit    EXPR GIVING NUM LEVELS
exit    0   (causes no exit)
```

iferror specifies the unit to execute via a -goto- if an error is found in the execution of a subsequent calculation in a unit

```
iferror UNIT NAME
iferror (B)  or  iferror q   (turns off -iferror- setting for
                              remainder of unit)
iferror EXPR,NAMEM,NAME0,q,NAME2,x   (example of conditional form;
           maximum of 100 arguments in conditional tag)
```

imain specifies the unit to execute at the start of every main unit in the lesson; later occurrence of the command overrides an earlier setting; equivalent to -do- at the beginning of each main unit

```
imain   UNIT NAME
imain   (B)  or  imain   q   (turns off -imain- setting for
                              remainder of lesson or until reset)
imain   EXPR,NAMEM,NAME0,q,NAME2,x   (example of conditional form;
           maximum of 100 arguments in conditional tag)
```

NOTE: The following two directives (-branch-, -doto-) are calc-type directives which permit branching or looping within a unit. When the directive is in the command field, it behaves like a -calc- command. In the tag field, the directive is part of a continued -calc-. In both cases commands which do not perform calculations are permitted within the -branch- or -doto- loop.

branch permits branching within a unit (the statement label must start with a number and may contain up to 7 characters)

for example:

```
5a      VAR⇐EXPR
.
.
.
branch  EXPR,5a,x      (argument x causes fall-through to the next
                        line in the unit)

calc    VAR⇐EXPR
6test   VAR⇐EXPR
        branch  6test
        branch  x    (causes fall-through to next line in the -calc-)
        branch  EXPR,x,6test     (example of conditional form)
```

doto permits looping within a unit (the statement label must start with a number, may contain up to 7 characters, and must have a blank tag)

for example:

```
doto    2sync,VAR⇐INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR
.
.
.
2sync   (B)

calc    VAR⇐EXPR
        doto  4run,VAR⇐INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR
                .
                .
                .
4run    (B)
```

Note: Stepsize, if omitted, is +1. Stepsize may be negative. Value of the loop variable is undefined after completion of the loop.

NOTE: The following four commands (-if-, -elseif-, -else-, and -endif-) permit branching within a unit. Logical value of an expression is "true" if its rounded value is -1 and "false" if its rounded value is 0.

if	performs a branch based on the logical value of the tag expression; value of "true" causes fall-through to the next line; value of "false" causes branch to the next -elseif-, -else-, or -endif- at the same level; code following -if- must be indented (up to the next -elseif-, -else-, or -endif- at the same level) and marked with the indent symbol; range of -if- must be terminated by -endif- at the same level `if      LOGICAL EXPR`
elseif	provides an alternative branch within the range of the preceding -if- at the same level; subsequent code follows same indenting rules as -if- `elseif  LOGICAL EXPR`
else	(no tag) provides a branch if the logical value of the tag of the preceding -if- or -elseif- at the same level is "false"; subsequent code follows same indenting rules as -if-
endif	(no tag) marks the end of the range of the preceding -if- at the same level

NOTE: Following is an example demonstrating placement of these commands.

```
if      n8<4
.       write   first branch          $$ executed if n8<4
.       calc    n9⇐ 34                $$ executed if n8<4
elseif  n8=4                          $$ executed if n8≥4
.       write   second branch         $$ executed if n8=4
.       do      someunit              $$ executed if n8=4
else                                  $$ executed if n8>4
.       write   default branch        $$ executed if n8>4
.       if      n8>6                  $$ executed if n8>4
.       .       write   special branch $$ executed if n8>6
.       endif                         $$ end of range of -if  n8>6-
endif                                 $$ end of range of -if  n8<4-
```

NOTE: The following four commands (-loop-, -endloop-, -outloop-, and -reloop-) permit looping within a unit. Logical value of an expression is "true" if its rounded value is -1 and "false" if its rounded value is 0.

loop initiates a loop based on the logical value of the tag expression; value of "true" causes execution of subsequent commands in the loop; value of "false" causes execution of the first command after -endloop- at the same level of indentation as -loop-; code following -loop- must be indented (up to the next -outloop-, -reloop-, or -endloop- at the same level) and marked with the indent symbol; range of -loop- is marked by -endloop- at the same level

```
loop    LOGICAL EXPR   (blank tag is equivalent to "true" value)
```

endloop (no tag) marks the end of a loop initiated by the previous -loop- command at the same level of indentation; causes a branch back to the previous -loop- command at the same level

outloop based on the logical value of the tag, causes exit from the range of -loop- at the same level of indentation; value of "true" causes execution of the first command after -endloop- at the same level; value of "false" causes execution of subsequent commands within the loop, which follow the same indenting rules as -loop-

```
outloop LOGICAL EXPR   (blank tag is equivalent to "true" value)
```

reloop based on the logical value of the tag expression, causes branch back to the previous -loop- command at the same level of indentation without terminating the loop; value of "true" causes branch to the previous -loop- at the same level; value of "false" causes execution of subsequent commands within the loop, which follow the same indenting rules as -loop-

```
reloop  LOGICAL EXPR   (blank tag is equivalent to "true" value)
```

Note: Following is an example demonstrating placement of these commands.

```
loop    n8<10
.       write   within loop         $$ executed if n8<10
.       subl    n8                  $$ executed if n8<10
reloop  n8≥5                        $$ executed if n8<10
.       write   still within loop   $$ executed if n8<5
.       do      someunit            $$ executed if n8<5
outloop n8<3                        $$ executed if n8<5
.       write   still within loop   $$ executed if 3≤n8<5
endloop
write   outside of loop             $$ executed if n8≥10 or n8<3
```

Key-initiated sequencing

NOTE: The following commands (-next- through -lablop-) may have the conditional form, where argument x leaves the pointer unchanged, and argument q clears the pointer and renders the key inactive (except for NEXT, which causes fall-through to the following unit). Argument q is not valid with -nextnow-. Up to 100 arguments are permitted in the conditional tag. The conditional expression is evaluated when the command is executed, not when the key is pressed.

next, next1, back, back1, stop specifies the unit executed when the student presses the appropriate key (arrows must be satisfied before sequencing on the NEXT key)

```
next    UNIT NAME
back1   UNIT NAME
back    (B)  or  back    q   (clears back pointer; disables BACK key)
```

nextnow terminates processing in the unit and makes only NEXT key active

```
nextnow UNIT NAME
```

nextop, nextlop, backop, backlop specifies the unit executed when the student presses the appropriate key; there is no full-screen erase and new information is plotted on-the-page; the unit specified, however, is a main unit

```
nextlop UNIT NAME
backop  (B)  or  backop  q   (clears back pointer; disables BACK key)
```

help, help1, data, data1, lab, lab1 initiates a help-type sequence by specifying the unit to be executed if the student presses the appropriate key; sets the base pointer for the unit to return to unless the base pointer is already set; the unit executed is a main unit but not a base unit (unless the base pointer is reset to this unit); a help-type sequence may be terminated by the -end- command

```
help    UNIT NAME
lab     (B)  or  lab     q   (clears lab pointer; disables LAB key)
```

helpop, helplop, dataop, datalop, labop, lablop specifies the unit executed when the student presses the appropriate key; the unit executed is not a main unit or a base unit and no full-screen erase is performed; control is returned to the main unit after execution of the helpop-type unit

```
helpop  UNIT NAME
dataop  (B) or dataop  q   (clears data pointer; disables DATA key)
```

term permits use of the TERM key to initiate a help-type sequence starting at the unit containing this command and the specified character string; sequence can be terminated by -end- (see -inhibit term-)

```
term    STRING   (maximum of 8 characters)
term    (B)   (provides match to any term request that does not
               match an author-specified or system-specified term)
```

termop similar to -term- except initiates a helpop-type sequence

```
termop  STRING   (maximum of 8 characters)
termop  (B)   (provides match to any term request that does not
               match an author-specified or system-specified term)
```

NOTE: A lesson may have up to 299 -term- and -termop- commands.

base resets or clears the base pointer in order to alter help-type sequencing

```
base    (B) or base    q   (clears base pointer)
base    UNIT NAME          (sets base pointer to named unit)
base    EXPR,q,NAME,x      (example of conditional form;
           argument x leaves base pointer unchanged; argument q
           clears base pointer; maximum of 100 arguments in
           conditional tag)
```

end terminates a help-type sequence or a lesson

```
end     (B) or end     help   (ends a help-type sequence; may
           occur anywhere in a unit; student is returned to the
           base unit after pressing NEXT; -end- is ignored in a
           non-help-type sequence)
end     lesson   (when NEXT is pressed after execution of this
           statement, the student is returned to the router lesson
           or to the "Press NEXT to Begin" page; finish unit is not
           executed; authors are returned to the author-mode page)
```

Timing

keylist (non-executable) forms a set of keys with the specified name for use with -pause- and -keytype- commands

```
keylist NAME,KEY1,KEY2,KEY3,...   (from 2 to 7 characters in NAME)
keylist NAME,NAME1,NAME2,...      (keylists may be combined)
```

Note: System-defined keylists are:

alpha	(letters: a to z and A to Z)
numeric	(digits: 0 to 9)
funct	(function keys ["key" from o200 to o235])
keyset	(any keyset input)
touch	(input from touch panel)
ext	(input from external device other than touch panel)
all	(input from keyset, touch-panel, or external device)

pause delays execution of subsequent commands by the specified interval or until the specified keys are pressed

```
pause   EXPR GIVING NUM SECONDS   (minimum of .75 second)
pause   0   (causes no pause; exception to .75 second minimum)
pause   (B)  or  pause   NEGATIVE VALUE   (interrupts processing
           until any keypress comes in)
pause   keys=KEY1,KEY2,KEYLIST NAME,...   (interrupts processing
           until one of the specified keys comes in; all keynames
           are typed without quote marks and function keys are
           typed in lower case)
pause   NUM SECONDS,keys=KEY1,KEY2,KEYLIST NAME,...   (interrupts
           processing for the specified time or until one of the
           specified keys comes in)
```

Note: If a function key other than next, such as help, is specified and there is a preceding -help- or -helpop- command specifying a unit to execute, this unit is executed rather than the command following the -pause-. If next is specified, the NEXT key just breaks the -pause-, even if there is a preceding -next- command.
The statements -pause keys=touch- and -pause keys=ext- set the appropriate -enable-.

collect allows storage of keycodes from keyset, touch panel, or external inputs in successive variables, starting at the specified variable; collection terminates with receipt of the specified number of keys or with receipt of the TIMEUP key, which is also stored

```
collect STARTING VAR,NUM KEYS   (must use student variables)
```

getcode stores a user-generated string, left-justified, in the specified variable and plots X's; "endkeys" specifies function keys which terminate the entry (in addition to NEXT, which is the default); up to 10 characters may be entered and stored

```
getcode VAR,endkeys=KEYNAME1,KEYNAME2,... (opt)  (names of keys are
                                                  in lower case)
```

keytype sets a variable according to the position in a list of the input by the user; if the input action is not listed, the variable is set to -1

```
keytype VAR,ARG0,ARG1,ARG2,...
```

arguments ARG0, ARG1, ARG2,... may be any of the following:

KEYNAME	(any keyname; no quotation marks are used; function keys are in lower case)
KEYLIST NAME	(name of a system-defined keylist or of a list set up by the -keylist- command)
(VAR)	(value of "key" is compared with the value stored in VAR)
ext(VAR)	(when the 10th bit from the right of "key" equals 1, indicating an external input, the right-most 9 bits of "key" are compared with the value stored in VAR)

touch(COARSE,WIDTH IN CHARACTERS,HEIGHT IN LINES)
touch(FINEX,FINEY,WIDTH IN DOTS,HEIGHT IN DOTS)

(COARSE or FINEX,FINEY is the screen position of the lower left corner of a rectangle with specified width and height; width and height are optional and are assumed to be 1 if omitted)

Note: Up to 100 keys may be specified; keylists count as one key.

time presses the TIMEUP key after the specified interval and sets "key" to "timeup"; function keys can break through the timing and set "key" to the key pressed

```
time    EXPR GIVING NUM SECONDS    (minimum of .75 second)
time    (B) or time    NEGATIVE VALUE  (clears any -time- in effect)
```

timel specifies a unit in the same lesson to execute (via helpop-type sequence) when the indicated time has elapsed; remains in effect across other timing commands and across unit boundaries

```
timel   NUM SECONDS,UNIT NAME    (minimum of .75 second)
timel   (B) or timel   NEGATIVE VALUE  (clears any -timel- in effect)
```

timer used in a router lesson to specify a unit in the router to which a routed student is sent when the indicated time has elapsed

```
timer   NUM SECONDS,UNIT NAME    (minimum of 60 seconds)
timer   (B) or timer   NEGATIVE VALUE  (clears any -timer- in effect)
```

press puts the specified key into the student input buffer for the indicated station, if given; limited to one keypress per second

```
press   KEYCODE
press   VAR CONTAINING KEYCODE
press   "KEYNAME"   (for non-function keys; lower case only)
press   KEYNAME     (for function keys, e.g., -press   next-)
press   KEYCODE,STATION   (presses the key at another station if the
           station is in the same lesson as the -press- command)
```

```
Note:  For 2-argument -press-:
       zreturn = -1  if the station is in the lesson
               =  0  otherwise
```

catchup (no tag) causes a pause in execution while transmission of accumulated output to the terminal is completed in order to synchronize display and execution of commands

break (no tag) interrupts processing and returns with a new timeslice for further processing when a complete timeslice is available

cpulim specifies the maximum CPU usage rate in thousand instructions per second with a maximum of 10 thousand instructions per second

```
cpulim  EXPR GIVING MAXIMUM CPU USAGE RATE   (maximum of 10)
```

Lesson connections and sections

use (non-executable) inserts into the file being condensed the specified block(s) from the file specified in the directory OR the file specified in the tag of -use-; all contiguous blocks with the same name are taken; use codewords on the files must match

use BLOCK NAME (file is specified in the directory)
use FILE NAME,BLOCK NAME (multi-file use flag is set in the directory; up to 5 different files may be used)

jumpout causes execution of the specified lesson or of the processor lesson, if one is declared; up to 10 arguments may be passed to the lesson (see -inhibit jumpchk-, -inhibit from-, and -args-)

jumpout FILE NAME (goes to the first unit in the lesson; jumpout codewords need not match)
jumpout FILE NAME(VALUE1,VALUE2) (example of form with arguments)
jumpout FILE NAME,UNIT NAME (goes to the specified unit in the lesson; jumpout codewords must match)
jumpout FILE NAME,UNIT NAME(VALUE) (example of form with arguments)
jumpout return (returns to the first unit of the lesson from which a jumpout was made to the present lesson)
jumpout return,return (returns to the lesson from which a jumpout was made to the unit following the unit with the -jumpout-)
jumpout (B) or jumpout q (causes a jumpout to the author-mode page for authors or to "Press NEXT to begin" page or to a router for students or instructors; similar to -end lesson-)
jumpout <LESLIST POSITION> (causes a jumpout to the first unit of the lesson at the specified position in the leslist)
jumpout <LESLIST POSITION>,UNIT NAME (causes a jumpout to the specified unit in the lesson at the specified leslist position; jumpout codewords must match)
jumpout EXPR;FILEM,UNITM;FILE0;FILE1,UNIT1;q;x (example of conditional form; argument q causes jumpout as above; argument x causes no jumpout)
jumpout resume (used in a router lesson to return the user to the lesson and unit specified by the last -restart-)
jumpout continue (used in a site lesson to send the user to a router or lesson)
jumpout NOTESFILE NAME (causes jumpout to the specified notes file; return to lesson is automatic, via -jumpout return,return-)
jumpout NOTESFILE NAME,datetime (causes jumpout to the specified notes file with date and time set in n2 and n3, respectively)
jumpout notes,choice (causes jumpout with read and write access to the student notes file attached to the user's group)
jumpout notes,read (causes jumpout with read access to the student notes file attached to the user's group)
jumpout notes,write (causes jumpout with write access to the student notes file attached to the user's group)
jumpout notes,instruct (causes jumpout with read access as an instructor to the student notes file attached to the group)
jumpout pnotes (causes jumpout to personal notes)

Note: Variable lesson and unit names must be enclosed in parentheses.

args stores the values of arguments passed by -jumpout- to the lesson; values are stored in the specified variables

```
args    VAR1,VAR2,VAR3   (maximum of 10 variables)

Note:  zreturn = -1  if arguments are passed successfully
               =  0  if no jumpout arguments are present
               ≥  1  if there are more jumpout arguments present than
                     variables in -args-; as many values as can be
                     received are stored; "zreturn" value is the
                     actual number of arguments present
```

Arguments passed by -jumpout- may be picked up by a -unit- command with arguments instead of by -args-.

from checks the lesson and main unit, if specified, from which a lesson is entered against a list and sets a variable to the relative position of the lesson and unit in the list; if no unit is specified, any unit in the lesson qualifies; if the lesson (and unit) are not listed, the variable is set to -1; alternate form stores lesson name and unit name in the specified variables

for example:

```
from    VAR;LESSON0,UNIT0;LESSON1;<LESLIST POSITION>,UNIT2
           (variable lesson and unit names must be enclosed in
           parentheses)
from    VAR FOR LESSON NAME,VAR FOR UNIT NAME (opt)
```

lessin checks if the lesson specified is credited to the user's logical site; sets "zreturn" to -1 if the lesson is in ECS and in use at the user's logical site and to 0 otherwise

```
lessin  'LESSON NAME'
lessin  (VAR CONTAINING LESSON NAME)
lessin  <LESLIST POSITION OF LESSON>
```

in sets "zreturn" to indicate whether a user at the specified station number is in the lesson containing the -in- command

```
in      EXPR GIVING STATION NUMBER

Note:  zreturn = -2  if the -in- command is in a router lesson and a
                     routed user at the specified station is in an
                     instructional lesson
               = -1  if the station is in the instructional lesson
                     containing the -in- command
               =  0  if the station is not in the lesson
```

notes initiates TERM-comments automatically, or sends the specified text to the lesson notes file or student notes file without user interaction; the title, if included, must be left-justified, may contain up to 15 characters, and always requires two variables

```
notes   STARTING VAR CONTAINING TITLE (opt)   (initiates TERM-comments)
notes   STARTING VAR CONTAINING TEXT,NUM VARS,STARTING VAR
        CONTAINING TITLE (opt)   (inserts text at front of note)
notes   STARTING VAR CONTAINING TEXT,NUM VARS,STARTING VAR
        CONTAINING TITLE (opt),send   (sends the text automatically)
```

Note: Student variables must be used for the text and the title; the format for the text is that for -text- command. After multi-argument -notes-, values of "zreturn" are:

```
zreturn = -1  if the note is sent successfully
        =  0  if the user pressed BACK1 and note was not sent
        =  1  if TERM-comments is not allowed in the lesson
        =  2  if the format of the text is incorrect or if the
              text is too long (>111 60-bit words or >16 lines)
        =  3  if the note cannot be stored (e.g., the notes file
              does not exist or is full)
```

cstart (non-executable) (no tag) indicates subsequent code is to be condensed (used after a preceding -cstop-)

cstop (non-executable) (no tag) indicates subsequent code is not to be condensed; in effect up to the next -cstart-, if any

cstop* (non-executable) (no tag) indicates none of the subsequent code is to be condensed, independent of subsequent -cstart- commands

NOTE: It is preferable to use the partial condense option of the editor rather than -cstart-, -cstop-, and -cstop*-.

Lesson lists

leslist references special blocks containing a list of up to 2400 lessons (numbered starting at 0); if the -leslist- command and leslist blocks are in different lessons, the common codewords must match or the codeword argument must be included; codeword argument must match the common codeword of the lesson containing the leslist blocks

```
leslist (0),LESLIST NAME
leslist (zlesson),LESLIST NAME
leslist ,LESLIST NAME
leslist LESLIST NAME
leslist LESSON NAME,LESLIST NAME,'CODEWORD' (opt)
          (LESSON NAME contains the leslist blocks)
leslist <LESLIST POSITION>,LESLIST NAME,'CODEWORD' (opt)
leslist (B)   (disconnects the current leslist)
```

Note: Variable arguments must be enclosed in parentheses. Quote marks on the codeword are omitted for variable argument.

```
zreturn = -1  if the -leslist- command is executed successfully
        =  0  if the leslist blocks are not found
        = +1  if codewords do not match
```

addlst allows addition of a lesson name to a leslist, either in the specified slot or in the first empty slot if none is specified; the tag must be a variable; requires three consecutive variables (the name is stored with: -storea STARTING VAR,30-)

```
addlst  STARTING VAR,LESLIST POSITION (opt)
```

```
Note:  zreturn = -1  if the lesson name is added successfully
               =  0  if there is no preceding successful -leslist-
                     command
               =  1  if the form of the lesson name is incorrect
               =  2  if the lesson name is already in the leslist
                     (with one-argument form only)
               =  3  if the leslist is full
               =  4  if the specified slot is occupied (with two-
                     argument form only)
               =  5  if the leslist is reserved by another user
```

removl allows deletion of a lesson at a specified leslist position; the vacated position is left blank

```
removl  LESLIST POSITION
```

```
Note:  zreturn = -1  if the lesson is removed successfully
               =  0  if there is no preceding successful -leslist-
                     command
               = +1  if the leslist is reserved by another user
```

reserve reserves the current leslist to prevent changes via -addlst- and -removl- by more than one user at a time

reserve leslist

Note: zreturn = -2 if the leslist is already reserved by this user
= -1 if -reserve- is executed successfully by this user
= 0 if there is no preceding successful -leslist- command
= 5+n, where n=station number of the user who has reserved the leslist

release releases the current leslist (if previously reserved)

release leslist

Note: zreturn = -2 if the leslist is not reserved by any user
= -1 if -release- is executed successfully by this user
= 0 if there is no preceding successful -leslist- command
= 5+n, where n=station number of the user who has reserved the leslist

lname stores the lesson name at the specified leslist position in three consecutive variables starting at the specified variable

lname STARTING VAR,LESLIST POSITION

Note: Use with -showa- to display the lesson name; e.g.:

```
lname   STARTING VAR,LESLIST POSITION
showa   STARTING VAR,30
```

zreturn = -1 if execution is successful
= 0 if there is no preceding sucessful -leslist- command

findl searches the leslist for the lesson name stored in three consecutive variables and returns the leslist position in the specified variable

findl STARTING VAR FOR LESSON NAME,VAR FOR LESLIST POSITION

Note: If the lesson name is not found or if no leslist is used, returned value for the leslist position is -1.

Lesson annotation and debugging

*****
c(space) indicates the statement on that line is a comment only and is to be ignored by the computer

```
*This is a comment.
c This is a comment.
```

$$ (not a command) when placed on the same line with a TUTOR statement indicates that subsequent material on that line is a comment

```
COMMAND TAG   $$this is a comment
```

change (non-executable) permits names of commands to be changed, e.g., to a language other than English; also permits symbols (e.g., punctuation) to be redefined in certain judging commands; -change- must be placed in the initial entry unit; all changes are in effect for the entire lesson and cannot be altered

```
change  command NORMAL TUTOR NAME to NEW NAME
change  symbol SYMBOL1 to SYMBOL2
```

for example:

```
change  command at to wo
change  symbol * to letter
change  symbol ? to puncword
change  symbol p to punc
change  symbol 3 to vowel
change  symbol a to b
change  symbol space to letter
change  symbol sup to null
change  symbol / to diacrit
    etc.
```

Note: The answer-matching commands affected by -change symbol- are: -answer-, -wrong-, -answerc-, -wrongc-, -concept-, -miscon-, -match-, -storen-. Other commands affected are: -getword-, -getmark-, -getloc-, and -compare-.

step allows a user to step through a lesson command by command; an author whose security code matches the lesson's change code may use TERM-step (TERM-step is not available for other user types)

```
step    on
step    off
step    EXPR   (value=0 turns off step; value≠0 turns on step)
```

*list (does not affect condensing or execution) specifies options for printing a file; -*list- commands for printing special types of blocks must precede those blocks in the program; source blocks and text blocks are always printed unless specified otherwise

*list	binary,BLOCK NAME,NUM WORDS,FORMAT	(prints contents of binary blocks; see page S20 for information on FORMAT)
*list	charset	(prints contents of all charsets in the lesson with O for dots on and - for dots off)
*list	charset,(DOTSBLANKS)	(prints contents of charsets with symbols specified for dots and blanks)
*list	commands,COMMAND1,COMMAND2,COMMAND3,...	(up to 10 commands; lists lines on print where specified commands appear)
*list	common,COMMON NAME,NUM WORDS,FORMAT	(prints contents of common; see page S20 for information on FORMAT)
*list	deleted	(prints deleted lines [with "mod words" option])
*list	eject	(causes page eject where command is located)
*list	ignore	(causes subsequent -*list- commands to be ignored)
*list	info	(prints lesson information display)
*list	label,YOUR LABEL INCLUDING SPACES	(prints a label at the location of the -*list- command)
*list	leslist	(prints the contents of all leslists in the lesson)
*list	listing	(prints the contents of listing blocks)
*list	micro	(prints the contents of all microtables in the lesson)
*list	mods	(prints "mod words": first 5 characters of the name of the last person to change each line and date of the change)
*list	nosource	(stops printing source blocks but not text blocks)
*list	notext	(stops printing text blocks but not source blocks)
*list	off	(stops printing blocks at the location of the command; starts printing preceding -*list- options and the unit cross-reference table)
*list	off,BLOCKNAME1,BLOCKNAME2,BLOCKNAME3-BLOCKNAME4	(specifies blocks that are not to be printed)
*list	parts	(prints only source blocks which are set to condense)
*list	symbols	(prints reference table of variables, defined and primitive, used in the lesson)
*list	text	(prints only text of -write- and -writec- commands)
*list	title,YOUR TITLE	(specifies subheading to be printed under the lesson name on each subsequent page; causes page eject when -*list title- is encountered)
*list	vocab	(prints contents of vocab blocks)
*list	xref,on	(turns on unit cross referencing [and symbol cross referencing if requested] for all printed source blocks; default case)
*list	xref,all	(same as -*list xref,on-)
*list	xref,off	(turns off cross referencing for all printed source blocks)
*list	xref,parts	(turns on unit cross referencing [and symbol cross referencing if requested] for all printed source blocks which are set to condense)
*list	1 space	(prints blocks with single spacing; default case)
*list	2 space	(prints blocks with double spacing)
*list	3 space	(prints blocks with triple spacing)
*list	4 space	(prints blocks with quadruple spacing)

Instructions for printing datasets and namesets are specified in the directory of the file. Page S20 has information on FORMAT.

datasets:

```
    STARTING RECORD NUMBER,NUM RECORDS,FORMAT
    STARTING RECORD,,special
    PAGE EJECTS;STARTING RECORD,,special
    PAGE EJECTS;special   (prints entire dataset in special format with page
                          ejects as specified)
    special               (prints entire dataset in special format)
    STARTING RECORD,,direct
    PAGE EJECTS;STARTING RECORD,,direct
    PAGE EJECTS;direct    (prints entire dataset in direct format with page
                          ejects as specified)
    direct                (prints entire dataset in direct format)
    INSTRUCTION1;INSTRUCTION2; etc.  (may state several different instructions)
```

namesets:

```
    NAMES;STARTING RECORD NUMBER,NUM RECORDS,FORMAT
    NAMES;STARTING RECORD,,special
    NAMES;PAGE EJECTS;STARTING RECORD,,special
    NAMES;PAGE EJECTS;special   (prints all records of specified names in
                                special format with specified page ejects)
    NAMES;special
    NAMES;STARTING RECORD,,direct
    NAMES;PAGE EJECTS;STARTING RECORD,,direct
    NAMES;PAGE EJECTS;direct
    NAMES;direct
    INSTRUCTION1;INSTRUCTION2; etc.  (may state several different instructions)
```

NOTE: PAGE EJECTS may be: pages and/or records (if both are given, the entries must be separated by a semicolon, e.g., pages;records).

With namesets, NAMES may be:

```
NAME1         (prints records only for NAME1)
NAME1-NAME2   (prints records for names from NAME1 to NAME2)
NAME1-        (prints records from NAME1 to the last name)
-NAME2        (prints records from the first name to NAME2)
omitted       (prints records for all names in the nameset; preceding
              semicolon must be included, e.g.,  ;special  or
              ;STARTING RECORD,,direct)
```

Note: FORMAT for printing datasets, namesets, commons, and binary blocks:

integer	or i	(nc-variables; prints 10 words per line)
exponential	or e	(vc-variables; prints 10 words per line)
floating	or f	(vc-variables; prints 10 words per line)
octal	or o	(prints 5 words per line)
hexadecimal	or h	(prints 6 words per line)
alpha	or a	(prints 10 words per line)
x		(prints each word in o, e, i, and a formats; prints 2 words per line)
special	or s	(special format; specified number of words to be printed is ignored but field must be present; details below)(not used with binary blocks)
direct	or d	(like special format but carriage control is required)
(DESIGNED)		(designed format; enclosed in parentheses; details below)

special format and direct format:

words are interpreted in alpha;
words with all 0 bits are ignored unless they are preceded by at least one non-zero word on the same line;
a line (and a page) must always end with at least 12 zero bits (o0000)
up to 127 characters may be printed per line;
control characters for direct: " ", single space; "0", double space; "-", triple space; "+", overwrite; "1", page eject; "2", bottom of page

the following print options are placed directly in the dataset, nameset, or common; each requires two consecutive words:

*format eject	(signals page eject at this location)
*format end	(indicates the print of the file or of the name is to end at this location)
*format pages	(signals page eject after each printed page; allows top and bottom margins)
*format records	(signals page eject after each subsequent record)
*format blocks	(signals page eject after each subsequent block)

designed format:

the format is for a line of print;
format must be enclosed in parentheses;
up to 135 characters may be printed per line

designed format may consist of:

i (integer; nc-variables; prints 10 characters per word)
e (exponential; vc-variables; prints 10 characters per word)
f (floating point; vc-variables; prints 10 characters per word)
o (octal; prints 20 characters per word)
h (hexadecimal; prints 15 characters per word)
a (alpha; prints 10 characters per word)
x (space; preceding number indicates number of spaces)
l (location of word; prints 4 or more characters)
p (skip to next word to be printed on the same line; preceding number indicates how many words to go forward)
commas and spaces for readability

System variables for sequencing

args	number of arguments transferred at the previous execution of a unit with arguments or -jumpout- with arguments
backout	= -2 for a single-station backout = -1 for a general backout = 0 for no backout (e.g. signoff via STOP1) = +1 after -station stop1-
baseu	name of the user's current base unit (= 0 if no base unit is specified, indicating the user is not in a help-type sequence)
clock	value of the system clock in seconds (to the nearest millisecond) since the previous deadstart (see command -clock-)
fromnum	leslist position of the lesson from which the user came via a jumpout (= -1 if the lesson is not in the leslist or if no leslist is in use)
key	after a keyset input: contains the 7-bit keycode of the last keypress; after a touch-panel input: contains a 9-bit number which gives the location of the touch square (the binary form of this number is 1xxxxyyyy, where the 4 bits labeled "x" give the horizontal touch location and the 4 bits labeled "y" give the vertical touch location--coordinates for touch squares on the screen are: 0,0 at lower left, 0,15 at upper left, 15,0 at lower right, 15,15 at upper right); after an external input: contains a 10-bit number whose left-most 2 bits are 10, with the remaining 8 bits carrying information from the external source
lessnum	leslist position of the user's current lesson (= -1 if the lesson is not in the leslist or if no leslist is being used)
lleslst	maximum number of lessons allowed in the leslist (= 0 if no leslist is in use)
llesson	condensed length of the lesson
mainu	name of the user's current main unit
mallot	memory allotment for the logical site at which the user is working
muse	total memory usage by users at the same logical site as the user

nhelpop	number of times a help-type key is pressed for on-the-page help; zeroed for each main unit and for each arrow in the unit
proctim	processing time in the lesson (in seconds, to nearest millisecond)
ptime	= -1 if the current time is during prime-time hours = 0 otherwise
sitenam	name of the user's logical site
station	identification number assigned by the system to a terminal physical site = station ars 5 = int(station/32) site station # = station $mask$ o37 = 32 × frac(station/32) station = 32 × physical site + site station #
tactive	number of currently active terminals
user	user type: 'author', 'instructor', 'student', 'multiple', 'sabort' (if student records have been aborted), 'snockpt' (if automatic checkpoint has been aborted)
usersin	number of users in the lesson (routed users are counted as being in the router as well as in the instructional lesson)
zaccnam	name of the account which contains the user's group
zcondok	= -1 if the lesson condenses without errors or warnings = 0 if the lesson has condense errors or warning messages
zfroml	name of the lesson from which a jumpout was done
zfromu	name of the unit from which a jumpout was done
zgroup	name of the user's group
zid	unique identification number for the user; information is in 3 fields (counting from the left end of the word): 18 bits: system identifier 22 bits: group identifier 20 bits: name identifier

zlesson	name of user's current lesson
zpnfile	= -1 if the user's group has a personal notes file attached = 0 otherwise (and for multiples and students without access to personal notes)
zpnotes	= -1 if the user has new, unread personal notes = 0 otherwise
zretrnu	name of the unit to which -jumpout return,return- will go
zreturn	set by some commands according to the results of execution; set by: TUTOR commands: -addlst-, -addname-, -addrecs-, -attach-, -charset-, -chartst-, -checkpt-, -commonx-, -comret-, -datain-, -dataout-, -delrecs-, -ext-, -in-, -getline-, -leslist-, -lessin-, -lineset-, -lname-, -micro-, -names-, -notes-, -parse-, -press-, -readd-, -readset-, -records-, -release-, -removl-, -rename-, -reserve-, -setline-, -setname-, -site-, -station-, -xin-, -xmit-, -xout-; µTUTOR and related commands: -attach-, -charset-, -compute-, -datain-, -dataout-, -loadu-, -putd-, -runu-, -trap-, -pptdata-, -pptload-, -pptout-, -pptrun-, -ppttest-
zsnfile	= -1 if student user's group has a student notes file attached (TERM-comments sent to student notes file) = 0 otherwise (and for authors and instructors) (TERM-comments sent to lesson notes file) = +1 if access to student notes and lesson notes is not allowed
zsnotes	= -1 if the student has new, unread notes = 0 otherwise
zsysid	60-bit value which uniquely represents the user's PLATO system; will not change during lifetime of the system; not for display purposes
zsystem	contains the name of the user's PLATO system
zterm	contains the last term requested by the user
ztouchx	fine-grid x-location of the center of the touch box touched (= -1 if last input was not a touch input)
ztouchy	fine-grid y-location of the center of the touch box touched (= -1 if last input was not a touch input)

ztzone	contains the three-letter abbreviation of the time zone of the location of the central computer of the user's PLATO system (e.g., CST)
zunit	name of the user's current unit
zusers	number of users currently signed on

NOTE: The following system variables contain alphabetic information (left-justified) and must be displayed with -showa-: baseu, mainu, sitenam, user, zaccnam, zfroml, zfromu, zgroup, zlesson, zretrnu, zsystem, zterm, ztzone, and zunit.

In addition to the system variables listed in this subsection, keynames of function keys may be treated as system constants. These keynames are typed in lower case (e.g., next, lab, term) and have the numerical values given in the keycode table on page A5. The exception is the SQUARE key, which has the keyname "microl".

This section presents commands and system variables available with the PLATO Programmable Terminal (PPT) and the CDC Information Systems Terminal (IST). Both types of terminals are referred to as "ppt". Details of assembly language are offered in several manuals and are not included here. Features of the TUTOR-like programming language (µTUTOR) and of central system TUTOR for running assembly language programs are described.

The µTUTOR language is evolving rapidly, and users should check "aids" for current features.

Loading and running

µtutor (non-executable) (no tag) marks the beginning of a terminal resident program written in µTUTOR; all subsequent commands are interpreted as µTUTOR commands; must follow the TUTOR part of a program

unit similar to the TUTOR -unit- but the contents of the µTUTOR unit are executed inside the terminal and must follow a -µtutor- command; form with arguments is not available

```
unit    NAME   (maximum of 8 characters in NAME)
```

loadu loads units into the terminal's memory so they can be executed with -runu- or referenced by sequencing commands in µTUTOR; must be placed in the TUTOR portion of a program

```
loadu   NAME1,NAME2,NAME3    (maximum of 20 units can be loaded)
loadu   ,NAME1,NAME2,NAME3   (loads specified units without deleting
                             those already loaded)
loadu   (B)   (clears flags indicating units are loaded)
```

```
Note:  zreturn = -1  if units are loaded successfully
               =  0  if the terminal is not programmable
               =  1  if there is not enough memory in the terminal
               =  2  if the units cannot be found
               =  3  if STOP or STOP1 is pressed during loading
               =  4  if µTUTOR is not available
               =  5  if there is a system error
               =  6  if too many units are loaded
```

runu causes execution of a unit which was previously loaded into the memory of the terminal (with -loadu-); must be placed in the TUTOR portion of a program

```
runu    NAME
```

```
Note:  zreturn = -1  if the -runu- is executed successfully
               =  0  if the specified unit is not loaded
```

haltu	(no tag) terminates execution of the unit which is executing when -haltu- is executed; if no unit is running, -haltu- is ignored; the µTUTOR unit must contain a -pause- command in order to be halted; -haltu- must be placed in the TUTOR portion of a program

Calculating

define (non-executable) similar to TUTOR -define- but primitives (n, v) are not used; names of variables, constants, arrays, and functions are listed, with the number of bits, if necessary; definitions are 16-bit signed integer type unless type is specified as floating point or 8-bit signed integer; all definitions following a specific type designation follow that designation until a different designation is encountered

for example:

```
define  NAME1,NAME2,NAME3
        NAME4(ARRAYSIZE)
        i,8:NAME5,NAME6
        FUNC(ARG1,ARG2)=EXPR
        f,48:NAME7,NAME8(ARRAYSIZE)
        i,16:NAME9
        i,8:NAME10=NAME9,NAME11=NAME9
        NAME12=20,NAME13=4.3
```

Note: Defined names may contain up to 7 characters and must start with a letter.
Up to 6 arguments are permitted in defined functions.
One-dimensional arrays are permitted.

calc similar to TUTOR -calc- (available functions are listed on page T7)

```
calc    VAR← EXPR
calc    VAR← "LETTER"   (single character only; character code is
           placed in the right-most 8 bits of VAR of integer type)
```

calcc similar to TUTOR -calcc-; the conditional expression must conform to µTUTOR restrictions on calculations

```
calcc   EXPR,VAR1← EXPRM,VAR2← EXPR0,VAR3← EXPR1,,VAR4← EXPR3
```

calcs similar to TUTOR -calcs-; the conditional expression must conform to µTUTOR restrictions on calculations

```
calcs   EXPR,VAR← EXPRM,EXPR0,EXPR1,EXPR2,,EXPR4
```

zero sets to zero a single variable or consecutive variables

```
zero    VAR
zero    STARTING VAR,NUM VARS
zero    (B)   (sets all defined variables to 0)
```

Note: In the 2-argument form, the number of bits zeroed is determined by the type designation of STARTING VAR.

set sets values of consecutive variables starting at the specified variable, or sets values of consecutive array elements starting at the specified element

```
set     STARTING VAR← EXPR1,EXPR2,EXPR3,...
```

compute evaluates a character string containing a simple expression involving constants and converts the string to a number

```
compute VAR FOR RESULT,STARTING VAR OF STRING,NUM CHARACTERS
```

```
Note:  zreturn = -1  if the string is converted successfully
               =  1  if the string contains an invalid character
               =  2  if there too many decimal points
               =  3  if the expression is too complicated
               =  4  if there is an unrecognized operator
               =  5  if the expression has bad form
               =  6  if there are unbalanced parentheses
```

randu selects a random number, sampled with replacement, and places it in the specified variable

```
randu   VAR,MAXIMUM
```

(selects integer from 1 to MAXIMUM; $0 \le$ MAXIMUM $\le (2^{15}-1)$)

Note: If the number generated is larger than the specified variable type can store, only the right-most bits are retained.

setperm creates a permutation list of the specified length for sampling by the -randp- command (similar to the two-argument -setperm- in TUTOR)

```
setperm LIST LENGTH,STARTING VAR OF LIST
```

(first variable of the list contains the number of integers remaining in the list; each remaining variable contains one bit for each integer)

randp selects an integer from the list set up by -setperm- and places it in the specified variable; when the list is exhausted, the variable is set to 0

```
randp   VAR FOR STORING VALUE,STARTING VAR OF PERMUTATION LIST
```

remove removes the specified value from the permutation list

```
remove  INTEGER TO REMOVE,STARTING VAR OF PERMUTATION LIST
```

restore restores the specified value to the permutation list

```
restore INTEGER TO RESTORE,STARTING VAR OF PERMUTATION LIST
```

block

copies consecutive variables from one location to another (similar to the TUTOR form of -block- except that there are no central memory variables)

```
block   FROM STARTING VAR,TO STARTING VAR,NUM VARS
```

Note: The number of 8-bit bytes copied is determined by the type designation of the "from" variable.

find

scans each variable in a list of consecutive variables for the first occurrence of the specified object

```
find    ARG1,ARG2,ARG3,ARG4
```

ARG1 = variable containing the object bit pattern
ARG2 = starting variable of the list (variables in the list must be the same type as the object)
ARG3 = number of variables in the list
ARG4 = variable for storing the location of the object (0 if found in first variable, 1 if found in second variable, etc., -1 if the object is not found)

pack

similar to TUTOR -pack- but packs each 6-bit character code into one 8-bit byte; string may contain embedded -show- and -showa-

```
pack    STARTING VAR FOR STORING STRING↕VAR FOR STORING
        CHARACTER COUNT (opt)↕STRING
```

search

searches a buffer for the first occurrence of the specified character string (each character occupies an 8-bit byte)

```
search  ARG1,ARG2,ARG3,ARG4,ARG5,ARG6
```

ARG1 = starting variable containing the object string to be searched for
ARG2 = number of 8-bit bytes in the object string
ARG3 = starting variable of the buffer to be searched
ARG4 = number of 8-bit bytes in the buffer to be searched
ARG5 = relative character position in the buffer at which to start the search
ARG6 = variable for storing the relative location of the object (0 if found in the first 8-bit byte, 1 if found in the second 8-bit byte, etc., -1 if not found)

searchf searches a buffer for the first occurrence of a character string in a specific field within an object

searchf ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7,ARG8

ARG1 = variable which contains the first character of the string to be found
ARG2 = number of 8-bit bytes in the string
ARG3 = starting variable of the buffer to be searched
ARG4 = number of entries in the buffer to be searched
ARG5 = entry in the buffer at which to start searching
ARG6 = number of 8-bit bytes in each entry in the buffer
ARG7 = starting byte position within each entry for comparison with the string
ARG8 = variable for storing the relative position of the object (0 if found in the first entry, 1 if found in the second entry, -1 if not found)

Operations, symbols, and functions used in calculations

addition	+
subtraction	-
multiplication	× or *
division	÷ or /
exponentiation	**
parentheses and brackets	( ), [ ], { }
assignment of a value to a variable	←
designation of an octal constant	o

° = degree sign; indicates a number is interpreted in degrees

π = pi = 3.14159...

e = 2.71828...

bit operations: $mask$, $union$, $diff$, ars, cls

logical operations: <, >, ≤, ≥, =, ≠

abs(X)	absolute value of X
int(X)	integer part of X
frac(X)	fractional part of X
log(X)	common logarithm of X (base 10)
alog(X)	common antilogarithm of X (i.e., 10^X)
ln(X)	natural logarithm of X (base e)
exp(X)	e^X
sin(X)	sine of X, X in radians; use sin(X°) for X in degrees
cos(X)	cosine of X, X in radians; use cos(X°) for X in degrees
comp(X)	one's complement of X (bit reversal)
zvloc(X)	absolute memory location in RAM of the variable X
zk(KEYNAME)	returns keyset code for KEYNAME (e.g., zk(m), which has value 77; KEYNAME must be specified; expression is not allowed; allowed keynames are given in the appendix, in the keycode table for programmable terminal)

Numbers are represented in "two's complement" form; i.e., -X = comp(X)+1 . Since integers have at most 16 bits and the left-most bit is the sign bit, the range of values for integers is -2^{15} to $+(2^{15} - 1)$, or -32768 to +32767 . (The larger range for negative values results from two's complement representation.)

Floating point numbers contain 48 bits: 1 bit for the sign bit, 15 bits for the exponent, and 32 bits for the coefficient. The range of values for floating point numbers is 2^{-16383} to 2^{+16383}.

Numbers have up to 9 significant digits.

File operations

attach establishes a connection between a μTUTOR lesson and a dataset which is stored on a floppy disk which is connected to the terminal

```
attach  NAME   (variable tag must be an integer variable and must be
               enclosed in parentheses)
```

```
Note:  zreturn = -1  if connection to the file is successful
               =  0  if the dataset is not found on the floppy disk
               = +1  if there is a disk error
```

datain transfers data from the dataset to the specified buffer

```
datain  STARTING RECORD NUMBER,TO STARTING VAR,NUM RECORDS
```

dataout transfers data from the specified buffer to the dataset

```
dataout STARTING RECORD NUMBER,FROM STARTING VAR,NUM RECORDS
```

NOTE: With -datain- and -dataout-:

```
zreturn = -1  if the transfer is successful
        =  0  if no dataset is attached
        = +1  if there is a disk error
```

Each record of a μTUTOR dataset contains 128 8-bit bytes. With -datain- and -dataout-, the receiving or sending buffer must accommodate the records received or sent.

Judging

darrow (non-executable) establishes a buffer (starting variable and length) for all subsequent -arrow- commands; if omitted, the buffer must be specified with the -arrow- command

```
darrow  STARTING VAR,NUM CHARACTERS ALLOWED
```

arrow places an arrow on the screen at the specified location and stores characters in the specified buffer; indented commands which follow -arrow- are executed before processing stops to wait for student input; non-indented commands which follow these indented commands but which precede response-matching commands are executed each time a judging key is pressed to initiate judging

```
arrow   LOCATION;STARTING VAR,NUM CHARACTERS ALLOWED
            (LOCATION may be COARSE  or  FINEX,FINEY)
arrow   LOCATION   (buffer established by preceding -darrow-)
```

endarrow (no tag) must terminate response processing; if the response is matched, indented commands following the matched response and indented commands following -ifmatch- are executed; if the "wrong" response is matched <u>or</u> if the response is not matched, judgment is "no" and processing stops until another response is entered; if judgment is "ok", response processing is complete and commands following -endarrow- are executed

long modifies the maximum number of characters allowed at an arrow set by the -arrow- command or the -darrow- command; must follow -arrow-; cleared at each -arrow-

```
long    NUM CHARACTERS
```

force initiates judging when the number of characters entered reaches the limit set by the most recent -long- command executed; must appear as an indented command following -arrow-; cleared at each -arrow-

```
force   long
```

jkey specifies keys (in addition to NEXT) which will initiate judging; must follow -arrow-; cleared at each -arrow-; if a non-function key is specified, it appears as the last key in the response buffer

```
jkey    KEY1,KEY2,KEY3  (e.g., jkey    back,=,a )
```

copy activates COPY key and specifies a buffer containing characters to be written on the screen one word at a time when COPY is pressed; the string is placed in the response buffer; must follow -arrow-; cleared at each -arrow-

```
copy    STARTING VAR,NUM CHARACTERS
```

putd replaces a character string in the response buffer with another character string; the first character in the tag is interpreted as the separator between strings; must follow -arrow-; cleared at each -arrow-

```
putd    /STRING1/STRING2/   (separator is /)
putd    ,STRING1,STRING2,   (separator is ,)

Note:  zreturn = -1  if -putd- is executed successfully
               =  0  if the replacement string causes the response to
                     be longer than the storage buffer
```

specs modifies standard judging procedures for all subsequent answer processing at that arrow; cleared at each -arrow-; settings are cumulative at an arrow; must follow -arrow-; cleared at each -arrow-

```
specs   nookno    (prevents appearance of "ok" and "no")
specs   okcap     (allows capitalized word in the response to match a
                  non-capitalized word in the tag of a response-
                  matching command)
specs   okextra   (allows "extra" words in the response, i.e., words
                  not in the tag of the response-matching command)
specs   okspell   (allows any reasonable spelling of words in the
                  response)
specs   punc      (allows only punctuation specified in the response-
                  matching command; without -specs  punc-, specified
                  punctuation must be present, but additional
                  punctuation may also be present)
specs   (B)       (clears previous settings at this arrow)
specs   nookno,okcap,okspell   (may combine tags)
```

NOTE: With the following response-matching commands (-keyword-, -answer-, -wrong-, -answerc-, -wrongc-, -exact-, -exactw-, -ansv-, -wrongv-), if the response matches the tag or the required argument, subsequent indented commands are executed up to the next non-indented command.

keyword checks the response for words listed in the tag; if a word is matched, the variable is set to the relative position of the matched word in the tag and judgment is "ok" ("zjudged" set to -1); if no word is matched, the variable is set to -1, judgment is not made, and judging continues

```
keyword VAR↕WORD0↕[WORD1 SYNONYM1]↕WORD2↕↕WORD4
```

answer compares the response with the -answer- tag; checks for spelling, capitalization, extra words, and punctuation unless altered by -specs-; punctuation marks are treated as words; sets "zjudged" to -1 if the response matches the tag

```
answer   <EXTRA WORDS>[SYNONYMS SEPARATED BY SPACES] WORD2 WORD3
answer   ❮a,STARTING VAR,NUM CHARACTERS❯    (maximum of 10 words)
```

wrong similar to -answer- but for an incorrect response; sets "zjudged" to 0 if the response matches the tag

```
wrong   <EXTRA WORDS>[SYNONYMS SEPARATED BY SPACES] WORD2 WORD3
wrong   ❮a,VAR,NUM CHARACTERS❯    (maximum of 10 words)
```

answerc conditional form of -answer-; performs checks available with -answer-; sets "zjudged" to -1 if the response matches the required argument

```
answerc EXPR↕RESPONSEM↕RESPONSE0↕↕RESPONSE2
```

wrongc similar to -answerc- but for an incorrect response; sets "zjudged" to 0 if the response matches the required argument

```
wrongc  EXPR↕RESPONSEM↕RESPONSE0↕RESPONSE1↕↕RESPONSE3
```

exact compares the response with the tag for an exact character by character match; sets "zjudged" to -1 if the response matches the tag

```
exact   STRING
```

exactw similar to -exact- but for an incorrect response; sets "zjudged" to 0 if the response matches the tag

```
exactw  STRING
```

ansv checks a numerical response against the first argument in the tag, with tolerance set by the optional second argument; sets "zjudged" to -1 if the response matches the tag within the tolerance; tolerance may be stated as absolute deviation or percent deviation; if tolerance is omitted, the response value must match the tag value

```
ansv    VALUE,TOLERANCE (opt)
```

wrongv similar to -ansv- but for an incorrect numerical response; sets "zjudged" to 0 if the response matches the tag within the tolerance

```
wrongv  VALUE,TOLERANCE (opt)
```

ok	judges a response "ok" and sets "zjudged" to -1 if the rounded value of the tag is negative; if the judgment is "ok", indented commands following -ok- are executed ok EXPR (blank tag is equivalent to negative value)
no	judges a response "no" and sets "zjudged" to +1 if the rounded value of the tag is negative; if the judgment is "no", indented commands following -no- are executed no EXPR (blank tag is equivalent to negative value)
or	(no tag) placed on the line between response-matching commands to provide alternative responses for the same value of "zanscnt"; if any tag of commands linked by -or- is matched, indented command following the last linked response-matching command are executed
ifmatch	(no tag) indented commands following -ifmatch- are executed whenever a response is matched, independent of judgment ("zjudged" equals -1, 0, or +1); only one -ifmatch- may occur for each -arrow-; -ifmatch- must be the last non-indented command before -endarrow-
iarrow	specifies the unit to be executed immediately after each subsequent -arrow- in a main unit; equivalent to indented -do- command after the -arrow- command; cleared at each main unit; later occurrence in the unit overrides an earlier setting in the unit iarrow UNIT NAME iarrow EXPR,UNITM,UNIT0,x,q,UNIT3 (example of conditional form) iarrow q (clears previous setting in the unit)
ijudge	specifies the unit to be executed each time the student presses a judging key; equivalent to non-indented -do- command after -arrow-following indented commands but preceding response-matching commands; cleared at each main unit; later occurrence in the unit overrides an earlier setting in the unit ijudge UNIT NAME ijudge EXPR,UNITM,UNIT0,q,UNIT2,x (example of conditional form) ijudge q (clears previous setting in the unit)

judge alters the judgment rendered by judging commands

judge	ok	(sets judgment to "ok"; sets "zjudged" to -1; processes subsequent regular commands before branching to -ifmatch- [or -endarrow-])
judge	no	(sets judgment to "no" [unanticipated]; sets "zjudged" to +1; processes subsequent regular commands before branching to -ifmatch- [or -endarrow-])
judge	wrong	(sets judgment to "no" [anticipated]; sets "zjudged" to 0; processes subsequent regular commands before branching to -ifmatch- [or -endarrow-])
judge	okquit	(sets judgment to "ok"; sets "zjudged" to -1; does not process subsequent regular commands; branches to -ifmatch- [or -endarrow-])
judge	noquit	(sets judgment to "no"; sets "zjudged" to +1; does not process subsequent regular commands; branches to -ifmatch- [or -endarrow-])
judge	quit	(does not alter judgment or "zjudged"; does not process subsequent regular commands; branches to -ifmatch- [or -endarrow-]; allows the student to leave the arrow even if judgment is not "ok")
judge	continue	(sets "zjudged" to 2; processes indented commands and then resumes processing non-indented judging commands)
judge	exdent	(sets "zjudged" to 2; does not process subsequent indented commands; processes non-indented judging commands)
judge	exit	(returns to the arrow to wait for additional input)
judge	ignore	(stops processing, erases response, and returns to the arrow to wait for additional input)
judge	x	(leaves judgment unchanged; used in conditional form)
judge	EXPR,no,ok,x,wrong	(example of conditional form)

getmark used after judging a response to give markup information on individual words in the response

getmark ARG1,ARG2

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable containing markup information
= -2 if the response is perfect or if no markup is done with the response-matching command used
= -1 if the position of the word is out of bounds (i.e., if ARG1>"zwcount")
= 0 if there are no errors in the word
> 0 bits in ARG2 are set according to the error(s), starting at the right-most bit (subscript "2" indicates the number is in binary notation):
(1_2) a word preceding this word is missing
(10_2) the word is out of order (too far right)
(100_2) there is a capitalization error
($1\ 000_2$) the spelling is incorrect
($10\ 000_2$) [bit not currently set]
($100\ 000_2$) the word is an extra word
($1\ 000\ 000_2$) this word is the last word, and a word which should follow is missing

getloc gives the screen position of the beginning (and end, if requested) of the specified word in the response

getloc ARG1,ARG2,ARG3,ARG4 (opt),ARG5 (opt)

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable for storing the finex screen position of the beginning of the word (= -1 if ARG1 > "wcount")
ARG3 = variable for storing the finey screen position of the beginning of the word
ARG4 = variable for storing the finex screen position of the end of the word (optional)
ARG5 = variable for storing the finey screen position of the end of the word (optional)

Presenting

write displays text, including embedded information

```
write   MESSAGE, INCLUDING EMBEDDED INFORMATION
```

writec displays one of several messages, depending on the value of the conditional expression; the conditional expression must conform to restrictions on calculations

```
writec  EXPR↕MESSAGEM↕MESSAGE0↕MESSAGE1↕↕MESSAGE3
```

NOTE: The following embed features are available. See descriptions of the individual commands for definitions of the arguments.

```
<show,EXPR>  or  <s,EXPR>
<showt,EXPR,LEFT,RIGHT>  or  <t,EXPR,LEFT,RIGHT>
<showb,EXPR,NUM BITS>    or  <b,EXPR,NUM BITS>
<showo,EXPR,NUM PLACES>  or  <o,EXPR,NUM PLACES>
<showh,EXPR,NUM PLACES>  or  <h,EXPR,NUM PLACES>
<showa,STARTING VAR,COUNT>  or  <a,STARTING VAR,COUNT>
<at,COARSE>; <at,FINEX,FINEY>
<atnm,COARSE>; <atnm,FINEX,FINEY>
```

show displays the value of an integer variable or expression

```
show    EXPR
```

showt displays the value of a variable or an expression in the specified format

```
showt   EXPR,PLACES LEFT OF DECIMAL,PLACES RIGHT OF DECIMAL
            (format, if omitted, is 4,3; if third argument is omitted,
            no places are shown to the right of the decimal)
```

showb displays the value of an integer variable or expression in binary notation; displays the specified number of bits, counting from the right end of the value

```
showb   EXPR,NUM BITS
```

showo displays the value of an integer variable or expression in octal notation; displays the specified number of places, counting from the right end of the value

```
showo   EXPR,NUM PLACES
```

showh displays the value of an integer variable or expression in hexadecimal notation; displays the specified number of places, counting from the right end of the value

```
showh   EXPR,NUM PLACES
```

showa displays characters in the specified integer variable(s), reading from the left end of the buffer; each character is in an 8-bit byte

```
showa   STARTING VAR,NUM CHARACTERS
```

erase erases the screen, selectively or entirely

```
erase   (B)   (causes full-screen erase)
erase   NUM CHARACTERS TO ERASE
erase   NUM CHARACTERS PER LINE,NUM LINES
```

mode specifies terminal writing mode

```
mode    write     (normal writing state; writes selected dots)
mode    erase     (erases selected dots)
mode    rewrite   (erases and rewrites in one step)
mode    inverse   (displays dark characters on light background)
mode    EXPR,erase,write,x,inverse   (example of conditional form;
           argument x leaves writing mode unchanged)
```

size specifies size bold-face or normal writing <u>or</u> sets size for -rdraw-

```
size    SIZE   (affects -rdraw- only)
size    bold   (specified bold-face writing)
size    (B)    (specifies normal writing; default condition)
```

rotate specifies vertical or normal writing <u>or</u> sets angle for -rdraw-

```
rotate  ANGLE IN DEGREES   (affects -rdraw- only)
rotate  vertical  (writing plots from bottom to top of screen)
rotate  (B)       (writing plots from left to right; default condition)
```

text displays contents of an alphanumeric buffer line by line; the end of a line must be indicated by an 8-bit byte equal to 0; not affected by -size- or -rotate-

```
text    STARTING VAR,NUM 8-BIT BYTES
```

textn similar to -text- except lines of text are numbered to the left of each line; not affected by -size- or -rotate-

```
textn   ARG1,ARG2,ARG3,ARG4,ARG5,ARG6

        ARG1 = starting variable of the buffer
        ARG2 = number of 8-bit bytes
        ARG3 = variable for storing the position of the next character
               to be plotted (1 + position of last character displayed)
        ARG4 = number of the first line displayed (if equal to 0, no
               text is displayed)
        ARG5 = number of the last line displayed (maximum is 31)
        ARG6 = maximum number of characters per line
```

charlim (non-executable) specifies the highest character number into which alternate font characters may be loaded by -char- or -charset-; if omitted, 128 slots are set aside in memory for storing characters

```
charlim NUMBER   (values from 0 to 127)
```

charset loads a character set into the terminal's memory from the floppy disk connected to the terminal

```
charset LESSON NAME,BLOCK NAME   (variable arguments must be enclosed
                                  in parentheses)
```

```
Note:  zreturn = -1  if the charset is loaded successfully
               =  0  if the charset is not found on the floppy disk
               = +1  if an error occurs in reading the floppy disk
```

char permits specification of specially designed characters for display

```
char    NAME,ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7,ARG8
char    NAME,STARTING VAR
```

Note: In the 9-argument form, ARG1 through ARG8 specify which of the 16 dots are lit in each of the 8 columns of the character. In the 2-argument form, STARTING VAR is the first of 8 consecutive 16-bit variables, each specifying the dots in each of the 8 columns, as in the 9-argument form.
NAME may be a character number or a defined name.

getchar copies the depiction of the specified character into the specified buffer (8 consecutive 16-bit integer variables or 16 consecutive 8-bit integer variables); one column of the character is in each 16 bits

```
getchar NAME,STARTING VAR
```

Note: NAME may be a character number or a defined name.

inhibit disables certain actions of µTUTOR in a unit; settings are cleared at each main unit and default settings are restored; effect within a unit is cumulative, i.e., later occurrence of -inhibit- is added to the effect of an earlier occurrence

inhibit arrow	(prevents plotting of the response arrow)
inhibit blanks	(prevents judging if a judging key is pressed before a response is entered; default setting)
inhibit erase	(prevents normal full-screen erase when proceeding to a new main unit)
inhibit keys	(prevents any keypress from breaking through -pause-)
inhibit plato	(prevents processing of output from the central system)
inhibit (B)	(establishes the default settings in this main unit; equivalent to: -allow arrow,erase,keys,plato- and -inhibit blanks-)

allow permits actions which have been inhibited in the unit by -inhibit-; effect within a unit is cumulative, i.e., later occurrence of -allow- is added to the effect of an earlier occurrence

allow arrow	(allows the response arrow to be plotted)
allow blanks	(allows null input at a response arrow; default is -inhibit blanks-)
allow erase	(allows a full-screen erase at a new main unit)
allow keys	(allows input from the keyset to break through -pause-)
allow plato	(allows processing of output from the central system)
allow (B)	(establishes settings opposite to default settings; equivalent to: -inhibit arrow,erase,keys,plato- and -allow blanks-)

xout sends data (in 8-bit bytes) contained in the specified variables to an external device; data is read starting with the left-most byte

```
xout    DEVICE ADDRESS,STARTING VAR,NUM 8-BIT BYTES TO SEND
```

xin collects data (in 8-bit bytes) from an external device and stores it in the specified variables; data is stored starting at the left-most 8-bit byte

```
xin     DEVICE ADDRESS,STARTING VAR,NUM 8-BIT BYTES TO STORE
```

NOTE: See descriptions in "aids" of TUTOR versions of -xout- and -xin- for current information on device addresses.

intrupt specifies a unit to execute (via -do-) when an interrupt is received from an external device

```
intrupt UNIT NAME
intrupt EXPR,NAMEM,NAME0,x,NAME2,q  (example of conditional form)
```

trap traps output from the central system in the specified buffer; executes the trapped output; traps status of the terminal

```
trap    save;STARTING VAR,NUM 8-BIT BYTES   (traps output)
trap    play;STARTING VAR,NUM 8-BIT BYTES   (executes trapped output)
trap    status;STARTING VAR,NUM 8-BIT BYTES  (saves terminal status
           seen by the central system; saves 27 bytes)
trap    terminal;STARTING VAR,NUM 8-BIT BYTES  (saves terminal status
           seen by the µTUTOR executor; saves 27 bytes)
trap    save,play;STARTING VAR,NUM 8-BIT BYTES
           (saves and executes trapped output)
```

Note: zreturn = number of 8-bit bytes trapped in the buffer

NOTE: The following commands have identical forms to their TUTOR counterparts: -at-, -atnm-, -circle-, -draw-, -dot-, -box-, -plot-, -fill-, -vector-, -enable-, -disable-, -play-, -record-, -slide-, -beep-;

graphing commands: -gorigin-, -axes-, -bounds-, -scalex-, -scaley-, -labelx, -labely-, -markx-, -marky-, -gdot-, -gdraw-, -gbox-, -gat-, -gatnm-, -vbar-, -hbar-, -gvector-.
With graphing commands these restrictions hold: -labelx- and -labely- require an explicit format for the labels in the format for -showt-. There are no default values for: -gorigin-, -axes-, -bounds-, -scalex-, and -scaley-. These parameters must be set explicitly.

relative commands: -rorigin-, -rat-, -ratnm-, -rdot-, -rdraw-.

Exchanging information with the central system

xmit permits exchange of data between the terminal and the central system; when -xmit- is in the TUTOR part of a program, data is sent to the terminal for use in a µTUTOR program; when -xmit- is in the µTUTOR part of a program, data is sent to the central system for processing in a TUTOR program

µTUTOR form (sends an 8-bit value to the central system, where it is processed as an external key, i.e., a key from an external device [to pick up data from a single -xmit-, -pause keys=ext- and processing of "key" must follow the -runu- command; to pick up data from any number of -xmit- commands, -enable ext- and -collect- must follow -runu-])

```
xmit    EXPR
```

TUTOR form (data is in the form of horizontal segments; byte size of data ≤ 16; number of 8-bit bytes received is in "zdata"; the data is picked up by the µTUTOR program by the -receive- command)

```
xmit    STARTING VAR,NUM SEGMENTS,SEGMENT SIZE (opt)
```

(SEGMENT SIZE, if omitted, is 60; if SEGMENT SIZE > 16, only the right-most 16 bits are sent)

Note: After the TUTOR -xmit-,

zreturn = -1 if the data is transmitted successfully
= 0 if no µTUTOR program is loaded
= +1 if the STOP key is pressed during transmission

receive collects data that is sent to the terminal from the central system (by -xmit-)

```
receive STARTING VAR,NUM 8-BIT BYTES
```

Note: If the byte size for data transmitted from the central system is > 8, the byte size for the receiving buffer must be 16.

sendkey (no tag) sends the current keyset key to the central system

Routing

lesson sets the system variable "zldone" to indicate whether a lesson is considered complete

```
lesson  complete     (sets "zldone" to -1)
lesson  incomplete   (sets "zldone" to 0)
lesson  no end       (sets "zldone" to +1)
lesson  EXPR,complete,incomplete,x,no end   (example of conditional
            form; argument x leaves "zldone" unchanged)
```

score places the value of the tag, rounded to the nearest integer, into the system variable "zscore"

```
score   EXPR   (value from 0 to 100)
score   (B)    (sets "zscore" to -1)
```

Sequencing

jumpn jumps to the specified unit but does not do any initializations, such as screen erase; clears the -do- stack

```
jumpn   UNIT NAME
```

jumpout causes immediate execution of the specified lesson on the floppy disk connected to the terminal

```
jumpout LESSON NAME   (variable tag must be an integer variable and
                      must be enclosed in parentheses)
jumpout (B)   (returns to the router lesson on the floppy disk)
```

press puts the specified key into the student input buffer

```
press   (zk(LETTER))
press   FUNCTION KEY  (use lower case, e.g., -press   next-)
press   (EXPR)
```

getkey (no tag) reads the next key from the key buffer and sets "zkey" to the value of the next key (sets "zkey" to -1 if the buffer is empty)

clrkey (no tag) clears the key buffer

NOTE: The following commands have identical forms to their TUTOR counterparts: -branch-, -if-, -elseif-, -else-, -endif-, -loop-, -reloop-, -outloop-, -endloop-, -cstart-, -cstop-, -cstop*-, -use-, -keytype-, -keylist-, -pause-.

The following commands have identical forms to their TUTOR counterparts with restrictions as noted:
-next-, -next1-, -back-, -back1-, -help-, -help1-, -data-, -data1-, -lab-, -lab1-, -stop-, -imain-, -goto-, -do-, -jump-, -base-, -doto-.

Argumented units are not available.
Conditional expressions must conform to restrictions on calculations.
There is no iterative -do-.
Tag "q" must be used to inactivate a key; blank tag is not available.
Explicit -next- command is required to proceed. (Automatic sequencing is not available.)
There is no -end- command; a help-type sequence ends with a unit with no -next- command.
Only the blank-tag form of -base- is available.
With -doto- all index values must be integers.

Running assembler programs

All commands in this subsection must be executed in a TUTOR program.

pptaddr establishes a base address or location in the read/write memory of the terminal for subsequent loading, testing, running, and clearing of a program or for loading of data

```
pptaddr ADDRESS
pptaddr ADDRESS FOR LOADING,ADDRESS FOR RELOCATING
pptaddr (B)   (sets base address to default, which is "ztbase"+2048)
```

pptload loads and relocates, if required, the program in the specified binary block at the location specified by the previous -pptaddr-; if LESSON NAME is omitted, the current lesson is assumed

```
pptload LESSON NAME (opt),BLOCK NAME
pptload (B)   (clears all flags indicating that programs are loaded
               into the terminal's memory)
```

```
Note:  zreturn = -1  if program is loaded successfully
               =  0  if the binary block is not found
               =  1  if the STOP key is pressed during loading
               =  2  if the terminal is not programmable
               =  3  if the binary has a bad length
               =  4  if there are too many programs in memory
               =  5  if there is an error in the binary format
               =  6  if there is a system disk error
```

ppttest sets "zreturn" to test whether the specified binary block is loaded in the terminal's memory at the address specified by a previous -pptaddr-; if LESSON NAME is omitted, the current lesson is assumed

```
ppttest LESSON NAME (opt),BLOCK NAME
```

```
Note:  zreturn = -1  if the binary is loaded at the previously
                     specified address
               =  0  if the binary is not loaded at that address
```

pptclr clears the flag indicating that the specified binary block is loaded in the terminal's memory at the address specified by a previous -pptaddr-, or clears flags for all binaries; if LESSON NAME is omitted, the current lesson is assumed

```
pptclr  LESSON NAME (opt),BLOCK NAME
pptclr  (B)   (clears all flags)
```

pptdata loads data from student variables or central memory variables into the terminal's memory, starting at the address specified by a previous -pptaddr-

```
pptdata STARTING VAR,NUM SEGMENTS,SEGMENT SIZE (opt)
          (SEGMENT SIZE, if omitted, is 60; if SEGMENT SIZE > 8,
          only the right-most 8 bits are sent)
```

```
Note:  zreturn = -1  if data is sent successfully
               =  0  if the STOP key is pressed during transmission
               = +1  if the terminal is not programmable
```

pptout sends data words and control words stored in student variables or central memory variables to the terminal; each "package" of data consists of a 19-bit word which specifies a terminal function; (LDE, or load echo, is not permitted)

```
pptout  STARTING VAR,NUM VARS (opt)  (NUM VARS, if omitted, is 1)
```

```
Note:  zreturn = -1  if data is sent successfully
               =  0  if the terminal is not programmable
```

Note: For a complete description of the data words and control words, see "The PLATO V Terminal" by Jack Stifle.

pptrun causes execution of the program residing at the address specified by a previous -pptaddr-, or specifies the entry to be executed in a jump table at the beginning of the program residing at the address specified by a previous -pptaddr- and sends an 18-bit data word, if requested

```
pptrun  (B)  (executes the program at the previously declared address)
pptrun  JUMP TABLE ENTRY,DATAWORD (opt)  (executes an instruction
          in a jump table; DATAWORD, if omitted, is 0)
```

```
Note:  zreturn = -1  if -pptrun- is successful
               =  0  if the terminal is not programmable
```

System variables for terminal resident processing

NOTE: System variables which appear only in a TUTOR program are indicated.

zanscnt	number of response-matching commands encountered at an arrow before the response is matched; = -1 if no tag is matched
zcomm	provides a counter for timing interrupts; incremented each time an interrupt is received
zdata	number of 8-bit bytes of data sent from the central system by -xmit-
zentire	= -1 if all required words are present in the response, = 0 otherwise
zextra	= -1 if there are no extra words in the response, = 0 otherwise
zjcount	number of characters entered at a μTUTOR -arrow-
zjudged	= -1 for any "ok" judgment = 0 for any "wrong" judgment (anticipated "no") = 1 for any "no" judgment (unanticipated "no") = 2 for a response which is not matched
zkey	contains the keycode of the last input (updated after -arrow-, -pause-, -getkey-, and at the beginning of a main unit)
zldone	= -1 if the user has encountered -lesson complete- = 0 if the user has encountered -lesson incomplete- = +1 if the user has encountered -lesson no end-
zntries	number of attempts at the current arrow
zorder	= -1 if the word order is correct, = 0 otherwise
zreturn	set by some commands according to the results of execution; set by: -attach-, -charset-, -compute-, -datain-, -dataout-, -loadu-, -putd-, -loadu-, -trap-, -pptdata-, -pptload-, -pptout-, -pptrun-, -ppttest-

zrouten	indicates entry conditions to the router lesson: = 0 if this is the first entry to the router lesson = 1 if this entry to the router is via -jumpout- = 2 if the router is returned to when the end of the instructional lesson is reached = 3 if the router is returned to when the instructional lesson is terminated by STOP1 keypress = 4 if the router is returned to when an execution error occurs in the instructional lesson
zscore	rounded value of the tag of µTUTOR -score- (value from 0 to 100)
zspell	= -1 if spelling is correct, = 0 otherwise
ztbase	(TUTOR) address of the first available read/write memory byte in the user's terminal
ztmem	(TUTOR) number of 8-bit bytes of memory for the user's terminal
ztmemr	(TUTOR) number of available 8-bit bytes in the terminal's memory after a -loadu- command
ztprog	(TUTOR) = -2 if the user's terminal is programmable and µTUTOR is available = -1 if user's terminal is programmable and µTUTOR is not available = 0 if user's terminal is not programmable
ztrap	number of bytes of output pending from the central system; each terminal word is 3 bytes, so "ztrap" is a multiple of 3
zttype	gives information on the user's terminal; counting from the left end of the 8-bit word: 1 bit: always 0 3 bits: memory configuration (see "aids" for details) 4 bits: terminal type = 0 if the terminal is a PLATO IV = 1 if the terminal is a PDP = 2 through 7 if the terminal is an IST (5 for IST2) = 8 through 10 if the terminal is a PPT (these values are subject to change) Note: The TUTOR version of "zttype" contains only the information on terminal type.
zwcount	number of words in the response (maximum of 50)

zwherex	fine-grid x location for the next display in μTUTOR processing
zwherey	fine-grid y location for the next display in μTUTOR processing

Additional notes on TERMINAL RESIDENT PROCESSING

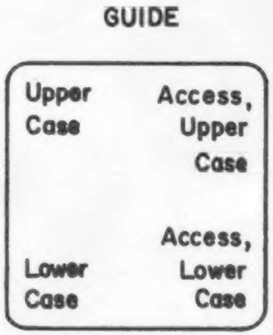
APPENDIX

Limits associated with commands (for TUTOR version unless specified otherwise)

area	10 characters in name of area
args argumented unit	10 arguments
arheada	five 6-bit characters
box gbox rbox	95 dots thick
bump	8 characters
calcc calcs	61 calculations
charlim	value of tag from 0 to 127
common	8000 variables (>1500 variables requires -comload-)
comload	1500 variables
compute	100 characters in character string
conditional form of commands which reference unit names, e.g., -next-, -help-, etc.	100 unit names (line containing 101st name is flagged as a condense error)
cpulim	value of tag from 1 to 10
define	7 characters in name of define set 7 characters in name of variable 1500 definitions (fewer if definitions are complicated) 500 definitions in define set "student" 6 arguments in defined function 10 units (e.g., kg, m, sec, liter, etc.) 255 elements in an array 5 active define sets
delay	1 second maximum
deletes	value of increment from 1 to 500
do join combined	10 levels
draw gdraw rdraw	63 arguments
edit	maximum of 300 characters in buffer

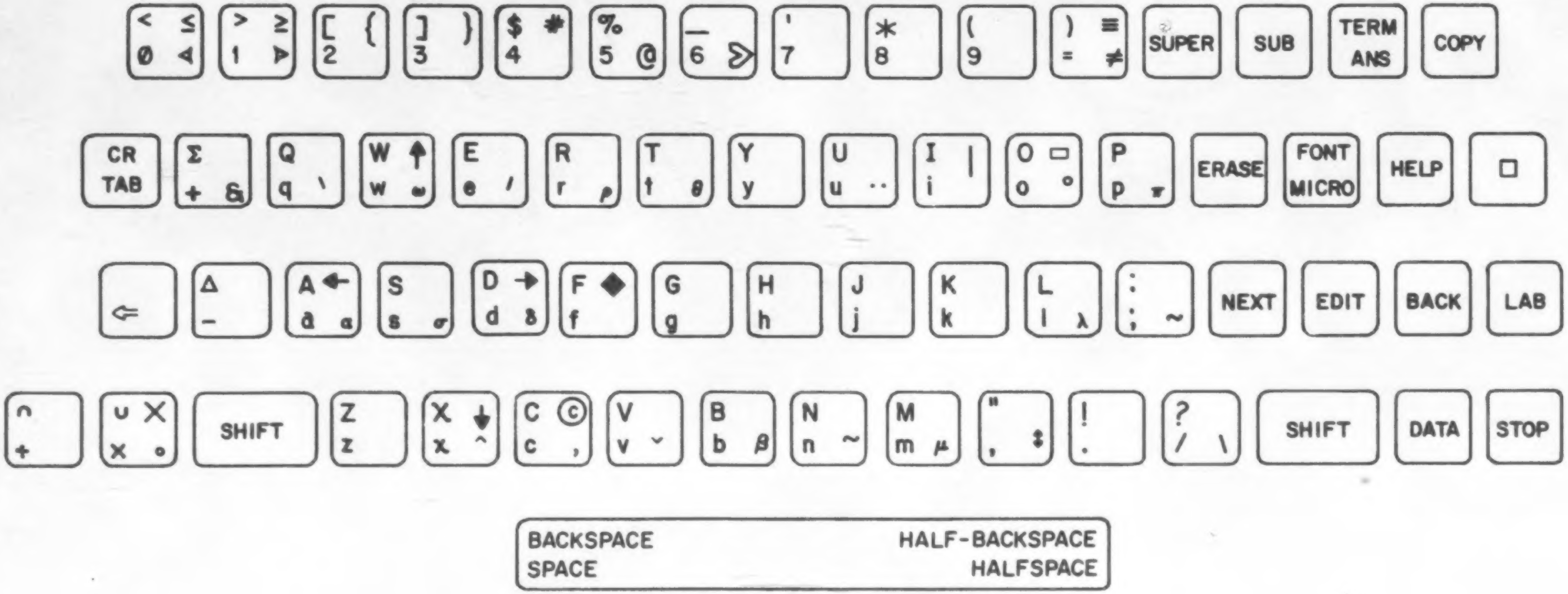
endings	10 separate -endings- commands, 8 endings in each tag
finds findsa	value of increment from 1 to 500
graph	9 characters in string to be plotted
group	8 characters in sign-on group name
hbar vbar	9 characters in string to be plotted
inserts	(increment between entries)×(number of entries to add) ≤ 500
keylist	from 2 to 7 characters in the name
keytype	100 keys in tag (keylists count as one key)
labelx labely markx marky	100 marks maximum plotted on an axis
list	7 characters in name of list
loada	300 characters
loadu	20 units
long	300 characters (>150 requires -edit- for active EDIT key)
lvars	128 local variables
move	5000 characters may be moved
name	18 characters in sign-on name
noword okword	9 characters in word
outputl	10 characters in label, 20 consecutive variables
pack	500 characters in character string (with embeds)
packc	100 character strings
pause	.75 second minimum
press	1 keypress per second
put putd putv	50 characters in strings; expansion due to string replacement limited to 300 characters

randu	integer from 1 to 2^{46} (TUTOR) integer from 1 to $(2^{15} - 1)$ (μTUTOR)
return	10 arguments
routvar	64 variables
score	value of tag from 0 to 100 (TUTOR and μTUTOR)
set	up to 61 separate values
setdat	value for "atime" less than elapsed time for the session values for "aarrows", "ahelp", etc. less than 512
setperm	(one-argument) integer from 0 to 120 (two-argument) integer from 0 to 3000
sort sorta	value of increment from 1 to 200
storage	8000 variables
term termop	8 characters in string, 299 terms in a lesson
time	.75 second minimum
timel	.75 second minimum
timer	60 seconds minimum
transfr	length is the smaller of: size of common or storage (reference to ECS) or length of -comload- or -stoload- (reference to CM) or 150 (reference to student variables) or tag of -routvar- (reference to router variables)
unit	8 characters in name
unitop	500 condensed words in a unit
entry	394 distinct condensed units in a lesson
vocab vocabs	7 characters in name of vocabulary
*list commands	10 commands


GUIDE
Upper Case
Access, Upper Case
Access, Lower Case
Lower Case

Standard PLATO Keyset

MICRO .	plots period + 7 spaces
MICRO Q	writes to left from current position
MICRO R	writes to right from current position
MICRO CR	writes to left in alternate font from right edge of screen
MICRO T	writes in boldface (thick) font (on ppt)
MICRO S	writes in standard font (on ppt)


< ≤ 0 ◀
> ≥ 1 ▶
[{ 2
] } 3
$ # 4
% 5 @
_ 6 ⋗
' 7
* 8
(9
) ≡ = ≠
SUPER
SUB
TERM ANS
COPY
CR TAB
Σ + &
Q q \
W ↑ w ω
E e /
R r ρ
T t θ
Y y
U u ¨
I i |
O ▭ o °
P p π
ERASE
FONT MICRO
HELP
□
⇦
Δ -
A ← a α
S s σ
D → d δ
F ◆ f
G g
H h
J j
K k
L l λ
: ; ~
NEXT
EDIT
BACK
LAB
∩ +
∪ × × ∘
SHIFT
Z z
X ↓ x ^
C © c ,
V v ˇ
B b β
N n ~
M m μ
" , ↕
! .
? / \
SHIFT
DATA
STOP
BACKSPACE
HALF-BACKSPACE
SPACE
HALFSPACE

KEYCODES[§]

key	keycode	key	keycode	key	keycode
a	1 = o01	,	46 = o56	_	97 = o141
b	2 = o02	.	47 = o57	'	98 = o142
c	3 = o03	÷	48 = o60	Σ	101 = o145
d	4 = o04	[	49 = o61	Δ	102 = o146
e	5 = o05	]	50 = o62	?	104 = o150
f	6 = o06	%	51 = o63	"	110 = o156
g	7 = o07	×	52 = o64	!	111 = o157
h	8 = o10	⇦	53 = o65	∩	112 = o160
i	9 = o11	sub	54 = o66	∪	116 = o164
j	10 = o12	super	55 = o67	locksub	118 = o166
k	11 = o13	shift*	56 = o70	locksup	119 = o167
l	12 = o14	car ret	57 = o71	:	127 = o177
m	13 = o15	<	58 = o72	FUNKEY	128 = o200
n	14 = o16	>	59 = o73	NEXT	130 = o202
o	15 = o17	bkspace	60 = o74	NEXT1	131 = o203
p	16 = o20	font	61 = o75	ERASE	132 = o204
q	17 = o21	access	62 = o76	ERASE1	133 = o205
r	18 = o22	;	63 = o77	HELP	134 = o206
s	19 = o23	A	65 = o101	HELP1	135 = o207
t	20 = o24	B	66 = o102	BACK	136 = o210
u	21 = o25	C	67 = o103	BACK1	137 = o211
v	22 = o26	D	68 = o104	LAB	138 = o212
w	23 = o27	E	69 = o105	LAB1	139 = o213
x	24 = o30	F	70 = o106	DATA	140 = o214
y	25 = o31	G	71 = o107	DATA1	141 = o215
z	26 = o32	H	72 = o110	TERM	142 = o216
0	27 = o33	I	73 = o111	ANS	143 = o217
1	28 = o34	J	74 = o112	COPY	144 = o220
2	29 = o35	K	75 = o113	COPY1	145 = o221
3	30 = o36	L	76 = o114	EDIT	146 = o222
4	31 = o37	M	77 = o115	EDIT1	147 = o223
5	32 = o40	N	78 = o116	MICRO	148 = o224
6	33 = o41	O	79 = o117	SQUARE	149 = o225
7	34 = o42	P	80 = o120	STOP	150 = o226
8	35 = o43	Q	81 = o121	STOP1	151 = o227
9	36 = o44	R	82 = o122	TAB	152 = o230
+	37 = o45	S	83 = o123	TIMEUP	155 = o233
-	38 = o46	T	84 = o124	CATCHUP	157 = o235
*	39 = o47	U	85 = o125	touch-	256 = o400
/	40 = o50	V	86 = o126	panel	↓ ↓
(	41 = o51	W	87 = o127	inputs	511 = o777
)	42 = o52	X	88 = o130	external	512 = o1000
$	43 = o53	Y	89 = o131	inputs	↓ ↓
=	44 = o54	Z	90 = o132		767 = o1377
space	45 = o55				

[§] "key" contains the keycode of the last key pressed.

INTERNAL CODES

unshifted		shifted		access		access-shifted	
char	code	char	code	char	code	char	code
a	o01	A	o7001	α	o7601	←	o767001
b	o02	B	o7002	β	o7602		
c	o03	C	o7003	¸	o7603	©	o767003
d	o04	D	o7004	§	o7604	→	o767004
e	o05	E	o7005	´	o7605		
f	o06	F	o7006			◆	o767006
g	o07	G	o7007				
h	o10	H	o7010				
i	o11	I	o7011			|	o767011
j	o12	J	o7012				
k	o13	K	o7013				
l	o14	L	o7014	λ	o7614		
m	o15	M	o7015	μ	o7615		
n	o16	N	o7016	~	o7616		
o	o17	O	o7017	°	o7617	□	o767017
p	o20	P	o7020	π	o7620		
q	o21	Q	o7021	`	o7621		
r	o22	R	o7022	ρ	o7622		
s	o23	S	o7023	σ	o7623		
t	o24	T	o7024	θ	o7624		
u	o25	U	o7025	¨	o7625		
v	o26	V	o7026	ˇ	o7626		
w	o27	W	o7027	ω	o7627	↑	o767027
x	o30	X	o7030	^	o7630	↓	o767030
y	o31	Y	o7031				
z	o32	Z	o7032				
0	o33	left64[ϕ]	o7033	‹	o7633		
1	o34	left[ϕ]	o7034	›	o7634		
2	o35						
3	o36						
4	o37						
5	o40	right[ϕ]	o7040	⊕	o7640		
6	o41	–	o7041	≥	o7641		
7	o42	'	o7042				
8	o43	bold[ϕ]	o7043				
9	o44	unbold[ϕ]	o7044				
+	o45	Σ	o7045	&	o7645		
-	o46	Δ	o7046				
*	o47						
/	o50	?	o7050	\	o7650		
(	o51						
)	o52			≡	o7652		
$	o53			✱	o7653		
=	o54			≠	o7654		
space	o55			.5space	o7655		

INTERNAL CODES (cont.)

unshifted		shifted		access		access-shifted	
char	code	char	code	char	code	char	code
,	o56	"	o7056	•	o7656		
.	o57	!	o7057				
÷	o60	^	o7060				
[	o61			{	o7661		
]	o62			}	o7662		
%	o63						
*	o64	˘	o7064	°	o7664	×	o767064
$	o65			cr 01[$]	o7665		
sub	o66	locksub	o7066	dnlin[$]	o7666	ldnlin[$]	o767066
super	o67	locksup	o7067	uplin[$]	o7667	luplin[$]	o767067
shift	o70						
car ret[$]	o71	ldnlin[$]	o7071	cr101[$]	o7671		
<	o72			≤	o7672		
>	o73			≥	o7673		
bkspace	o74			.5bksp	o7674		
font	o75						
access	o76						
;	o77	:	o7077	~	o7677		

[$] NOTE:
left64: write to left, starting at character position 64
left: write to left, starting at "where" position
right: write to right, starting at "where" position
bold: write in bold characters (ppt only)
unbold: write in standard characters (ppt only)
cr 01: carriage return to next line, character position 1
dnlin: write next character down one line (down 16 dots)
ldnlin: same as dnlin except locking
uplin: write next character up one line (up 16 dots)
luplin: same as uplin except locking
car ret: standard carriage return to next line, left margin
cr101: carriage return to screen position 101

The following characters, although produced with the shift key, do not produce a shift code in the internal code: *, (,), $, [,], %, <, >, car ret, backspace, and font. In this table these characters are in the "unshifted" category.

ALTERNATE FONT TERMINAL MEMORY LOCATIONS

(key associated with terminal memory location)

loc	key	loc	key	loc	key	loc	key	loc	key
0	space	27	0	54	(k)	81	Q	108	)
1	a	28	1	55	(l)	82	R	109	(E)
2	b	29	2	56	(m)	83	S	110	(F)
3	c	30	3	57	(n)	84	T	111	(G)
4	d	31	4	58	(o)	85	U	112	ᴖ
5	e	32	5	59	;	86	V	113	(H)
6	f	33	6	60	"	87	W	114	(I)
7	g	34	7	61	,	88	X	115	(J)
8	h	35	8	62	.	89	Y	116	ᴗ
9	i	36	9	63	unav	90	Z	117	(/)
10	j	37	+	64	space	91	<	118	(K)
11	k	38	-	65	A	92	>	119	(L)
12	l	39	(a)	66	B	93	[	120	(M)
13	m	40	/	67	C	94	]	121	(N)
14	n	41	(b)	68	D	95	$	122	(O)
15	o	42	(c)	69	E	96	%	123	:
16	p	43	(d)	70	F	97	_	124	!
17	q	44	⇒	71	G	98	'	125	(!)
18	r	45	(e)	72	H	99	*	126	(.)
19	s	46	(f)	73	I	100	(	127	unav
20	t	47	(g)	74	J	101	Σ		
21	u	48	÷	75	K	102	Δ		
22	v	49	(h)	76	L	103	(A)		
23	w	50	(i)	77	M	104	?		
24	x	51	(j)	78	N	105	(B)		
25	y	52	×	79	O	106	(C)		
26	z	53	⇎	80	P	107	(D)		

NOTE: unav: Location is unavailable on standard terminals.
(key): Press MICRO followed by the indicated key.

KEYCODES[3]
(programmable terminal)

key	keycode	key	keycode	key	keycode	key	keycode
timeup	-1						
0	0	<	32	space	64	bksp	96
1	1	>	33	a	65	A	97
2	2	[	34	b	66	B	98
3	3	]	35	c	67	C	99
4	4	$	36	d	68	D	100
5	5	%	37	e	69	E	101
6	6	_	38	f	70	F	102
7	7	'	39	g	71	G	103
8	8	*	40	h	72	H	104
9	9	(	41	i	73	I	105
×	10	∪	42	j	74	J	106
÷	11	∩	43	k	75	K	107
tab	12	cr	44	l	76	L	108
assign	13	assign1	45	m	77	M	109
+	14	Σ	46	n	78	N	110
-	15	Δ	47	o	79	O	111
sup	16	sup1	48	p	80	P	112
sub	17	sub1	49	q	81	Q	113
ans	18	term	50	r	82	R	114
erase	19	erase1	51	s	83	S	115
micro	20	font	52	t	84	T	116
help	21	help1	53	u	85	U	117
next	22	next1	54	v	86	V	118
edit	23	edit1	55	w	87	W	119
back	24	back1	56	x	88	X	120
data	25	data1	57	y	89	Y	121
stop	26	stop1	58	z	90	Z	122
copy	27	copy1	59	=	91	)	123
square	28	square1	60	;	92	:	124
lab	29	lab1	61	/	93	?	125
				.	94	!	126
				,	95	"	127

[3] System variable "zkey" contains the keycode of the last input.
Function "zk" returns keyset keycode values.
Touch-panel input codes range from 256 to 511.
External input codes range from 512 to 767.

Powers of 2

n	2^n		n	2^n	
0	1	$=8^0$	30	1 073 741 824	$=8^{10}$
1	2		31	2 147 483 648	
2	4		32	4 294 967 296	
3	8	$=8^1$	33	8 589 934 592	$=8^{11}$
4	16		34	17 179 869 184	
5	32		35	34 359 738 368	
6	64	$=8^2$	36	68 719 476 736	$=8^{12}$
7	128		37	137 438 953 472	
8	256		38	274 877 906 944	
9	512	$=8^3$	39	549 755 813 888	$=8^{13}$
10	1 024		40	1 099 511 627 776	
11	2 048		41	2 199 023 255 552	
12	4 096	$=8^4$	42	4 398 046 511 104	$=8^{14}$
13	8 192		43	8 796 093 022 208	
14	16 384		44	17 592 186 044 416	
15	32 768	$=8^5$	45	35 184 372 088 832	$=8^{15}$
16	65 536		46	70 368 744 177 664	
17	131 072		47	140 737 488 355 328	
18	262 144	$=8^6$	48	281 474 976 710 656	$=8^{16}$
19	524 288		49	562 949 953 421 312	
20	1 048 576		50	1 125 899 906 842 624	
21	2 097 152	$=8^7$	51	2 251 799 813 685 248	$=8^{17}$
22	4 194 304		52	4 503 599 627 370 496	
23	8 388 608		53	9 007 199 254 740 992	
24	16 777 216	$=8^8$	54	18 014 398 509 481 984	$=8^{18}$
25	33 554 432		55	36 028 797 018 963 968	
26	67 108 864		56	72 057 594 037 927 936	
27	134 217 728	$=8^9$	57	144 115 188 075 855 872	$=8^{19}$
28	268 435 456		58	288 230 376 151 711 744	
29	536 870 912		59	576 460 752 303 423 488	

Given the byte size = n: range for unsigned integers is 0 to 2^n-1
range for signed integers is $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

Given the maximum absolute value such that $2^{n-1} \leq |\text{maximum}| < 2^n$:
byte size for unsigned integers is n
byte size for signed integers is n+1

Alphabetical index to system variables

System variables may be used wherever expressions are accepted, e.g., in tag of -calc-, -at-, etc.

Word	Page
aarea	D6
aarrows	D6
ahelp	D6
ahelpn	D6
anscnt	J19
ansok	J19
aok	D6
aokist	D6
args	S21
asno	D6
aterm	D6
atermn	D6
atime	D7
auno	D7
backout	S21
baseu	S21
capital	J19
clock	S21
dataon	D6
entire	J19
errtype	R3
extra	J20
formok	J21
fromnum	S21
jcount	J19
judged	J19
key	S21
lcommon	C22
ldone	R3
lessnum	S21
lleslst	S21
llesson	S21
lscore	R3
lstatus	R3
lstorag	C22
mainu	S21
mallot	S21
mode	P21
muse	S21
nhelpop	S22
ntries	J19
opcnt	J20
order	J20
phrase	J20
proctim	S22
ptime	S22
rcallow	R3
router	R3
rstartl	R3
rstartu	R3
rvallow	R3
sitenam	S22
size	P21
sizex	P21
sizey	P21
spell	J20
station	S22
tactive	S22
user	S22
usersin	S22
varcnt	J20
vocab	J20
wcount	J20
where	P21
wherex	P21
wherey	P21
zaccnam	S22
zanscnt	T25
zbpc	C22
zbpw	C22
zcheck	F18
zcomm	T25
zcondok	S22
zcpw	C22
zcusers	C22
zdata	T25
zentire	T25
zextra	T25
zfacc	F18
zfile	F18
zfroml	S22
zfromu	S22
zftype	F18
zfusers	F18
zgroup	S22
zid	S22
zinfo	F18
zjcount	T25
zjudged	T25
zkey	T25
zlang	P21
zldone	T25
zleserr	R4
zlesson	S23
zline	F18
znindex	F18
znscpn	F18
znsmaxn	F18
znsmaxr	F18
znsnams	F18
znsrecs	F19
zntries	T25
zorder	T25
zpnfile	S23
zpnotes	S23
zrecs	F19
zretrnu	S23
zreturn	S23
zroff	F19
zrouten	T26
zrstatn	F19
zrtype	F19
zrvars	F19
zrvret	F19
zscore	T26
zsessda	D6
zsesset	D6
zsesspt	D6
zsnfile	S23
zsnotes	S23
zspell	T26
zsvars	F19
zsvret	F19
zsysid	S23
zsystem	S23
ztbase	T26
zterm	S23
ztmem	T26
ztmemr	T26
ztouchx	S23
ztouchy	S23
ztprog	T26
ztrap	T26
zttype	T26
ztzone	S24
zunit	S24
zusers	S24
zwcount	T26
zwherex	T27
zwherey	T27
zwpb	F19
zwpr	F19
zxfile	F19

Alphabetical index to commands and related directives

Command	Page
abort	C19
addlst	S15
addl	C3
addname	F4
addrecs	F5
allow	R1
allow	T18
altfont	P20
ans	J12
ansu	J11
ansv	J11,T11
answer	J9,T11
answerc	J10,T11
array	C1
area	D2
args	S13
argument	S1,S2
arheada	J2
arrow	J2,T9
arrowa	J2
at	P1,T19
atnm	P1,T19
attach	F1,T8
audio	P16
axes	P10,T19
back	S7,T22
backl	S7,T22
backgnd	C21
backop	S7
backlop	S7
base	S8,T22
beep	P17,T19
block	C18,T5
bounds	P10,T19
box	P6,T19
branch	S4,T22
break	S11
bump	J5
c	S17
calc	C3,T3
calcc	C3,T3
calcs	C3,T3
catchup	S11
change	S17
char	P5,T17
charlim	T17
charset	P19,T17
chartst	P20
checkpt	F13
circle	P7,T19
circleb	P7
clean	C13
clock	C10
close	J5
clrkey	T22
collect	S9
comload	C19
common	C19
commonx	C20
compare	J14
comret	C19
compute	C10,T4
concept	J10
copy	J3,T10
cpulim	S11
cstart	S14,T22
cstop	S14,T22
cstop*	S14,T22
darrow	T9
data	S7,T22
datal	S7,T22
datain	F2,T8
dataoff	D1
dataon	D1
dataop	S8
datalop	S8
dataout	F2,T8
date	C10
day	C10
define	C1,T3
delay	P4
deletes	C16
delname	F5
delrecs	F5
delta	P15
detach	F1
disable	P16,T19
do	S2,T22
dot	P6,T19
doto	S4,T20
draw	P6,T19
edit	J2
else	S5,T22
elseif	S5,T22
embed	P1,T15
enable	P16,T19
end	S8
endarrow	J3,T9
endif	S5,T22
endings	J4
endloop	S6,T22
entry	S1
erase	P3,T16
eraseu	J1
exact	J10,T11
exactc	J10
exactv	J11
exactw	T11
exit	S3
ext	P17
extout	P17
fill	P6,T19
find	C17,T5
findall	C17
findl	S16
finds	C15
findsa	C15
finish	D5
force	J1,T9
foregnd	C21
from	S13
funct	P15
gat	P12,T19
gatnm	P12,T19
gbox	P13,I19
gcircle	P13
gdot	P13,T19
gdraw	P13,T19
getchar	T17
getcode	S10
getkey	T22
getline	F16
getloc	J14,T14
getmark	J13,T14
getname	F4,F14
getword	J13
gorigin	P10,T19
goto	S2,T22
graph	P13
group	C10
gvector	P14,T19
haltu	T2
hbar	P14,T19
help	S7,T22
helpl	S7,T22
helpop	S8
helplop	S8
hidden	P3
htoa	C12
iarrow	J16,T12
iarrowa	J16
ieu	S1
if	S5,T22
iferror	S3
ifmatch	T12
ignore	J15
ijudge	T12
imain	S3,T22
in	S13
inhibit	P5,T18
initial	C20
inserts	C16
intrupt	T19
iospecs	F15
itoa	C12
jkey	J2,T9
join	J16,S3
judge	J17,T13
jump	S2,T22
jumpn	T22
jumpout	S12,T22
keylist	S9,T22
keytype	S10,T22
keyword	T10
lab	S7,T22
labl	S7,T22
labelx	P11,T19
labely	P11,T19
labop	S8
lablop	S8
lang	P3
leslist	S15
lessin	S13
lesson	R2,T21
lineset	P20
list	J4
lname	S16
loada	J5
loadu	T1
long	J2,T9
loop	S6,T22
lscalex	P11
lscaley	P11
lvars	C3
markup	J18
markupy	J18
markx	P11,T19
marky	P11,T19
match	J9
micro	P19

Alphabetical index to commands and related directives (cont.)

Command	Page
miscon	J10
mode	P4,T16
modperm	C9
move	C12
name	C10
names	F6,F15
next	S7,T22
nextl	S7,T22
nextnow	S7
nextop	S7
nextlop	S7
no	J15,T12
notes	S14
noword	J18
ok	J15,T12
okword	J18
open	J8
or	J12,T12
otoa	C12
outloop	S6,T22
output	D2
outputl	D2
pack	C11,T5
packc	C12
parse	F17
pause	S9,T22
permit	D5
play	P16,T19
plot	P5,T19
polar	P12
pptaddr	T23
pptclr	T23
pptdata	T24
pptload	T23
pptout	T24
pptrun	T24
ppttest	T23
press	S11,T22
put	J5
putd	J5,T10
putv	J5
randp	C8,T4
randu	C8,T4
rat	P8,T19
ratnm	P8,T19
rbox	P8
rcircle	P9
rdot	P8,T19
rdraw	P8,T19
readd	D3
readset	D3
receive	T20
recname	C13
record	P16,T19
records	F7
release	C21,F3,S16
reloop	S6,T22
remove	C9,T4
removl	S15
rename	F4
reserve	C21,F2,S16
restart	D5
restore	C9,T4
return	S3
rorigin	P8,T19
rotate	P4,T16
route	R1
routvar	R1
runu	T1
rvector	P9
say	P18
sayc	P18
saylang	P18
scalex	P10,T19
scaley	P10,T19
score	R2,T21
search	C11,T5
searchf	T6
seed	C8
segment	C1
segmentf	C1
sendkey	T20
set	C4,T4
setdat	D2
setline	F16
setname	F3,F14
setperm	C8,T4
show	P2,T15
showa	P3,T16
showb	T15
showe	P2
showh	P3,T16
showo	P3,T15
showt	P2,T15
showz	P2
site	M1
size	P4,T16
slide	P16,T19
sort	C14
sorta	C14
specs	J6,T10
station	M3
status	R2
step	S17
stoload	C21
stop	S7,T22
storage	C20
store	J8
storea	J8
storen	J8
storeu	J8
subl	C4
tabset	P19
term	S8
termop	S8
text	P3,T16
textn	T17
time	S10
timel	S10
timer	S10
touch	J12
touchw	J12
transfr	C18
trap	T19
unit	S1,T1
unitop	S1
use	S12,T22
vbar	P14,T19
vector	P7,T19
vocab	J4
vocabs	J4
window	P7
write	P1,T15
writec	P1,T15
wrong	J9,T11
wrongc	J10,T11
wrongu	J11
wrongv	J11,T11
xin	P17,T18
xmit	T20
xout	P17,T18
zero	C4,T3
*	S17
$$	S17
*format	S20
*list	S18
µtutor	T1

C
D
F
J
M
P
R
S
T
A
I